UNIVERZA V LJUBLJANI
FAKULTETA ZA RAČUNALNIŠTVO IN INFORMATIKO

Miha Grohar

Integracija hibridnega oblaka z virtualnim laboratorijem

DIPLOMSKO DELO
NA UNIVERZITETNEM ŠTUDIJU

Mentor: doc. dr. Mojca Ciglarič

Ljubljana, 2011

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01736/2011
Datum: 15.03.2011

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **MIHA GROHAR**

Naslov: **INTEGRACIJA HIBRIDNEGA OBLAKA Z VIRTUALNIM LABORATORIJEM**

INTEGRATING HYBRID CLOUD WITH VIRTUAL COMPUTING LABORATORY

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Opišite problem pomanjkanja virov ob konicah v infrastrukturi virtualnega laboratorija LRK VCL. Preučite možnosti rešitve z uporabo hibridnih oblakov. Pojasnite pojem javnega, zasebnega in hibridnega oblaka in definirajte funkcionalnosti vmesnika, ki bo povezoval zasebni oblak LRK VCL z javnim oblakom Amazon EC2. Načrtovani vmesnik implementirajte in preizkusite v testnem in produkcijskem okolju. Rešitev ovrednotite z vidika varnosti in uporabnosti ter navedite možnosti za nadaljnji razvoj.

Mentor:

doc. dr. Mojca Ciglarič

M. Ciglarič

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Dekan:

prof. dr. Nikolaj Zimic

IZJAVA O AVTORSTVU

diplomskega dela

Spodaj podpisani Miha Grohar,

z vpisno številko 63040045,

sem avtor diplomskega dela z naslovom: Integracija hibridnega oblaka z virtualnim laboratorijem

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mojce Ciglarič
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, dne 15.09.2011 Podpis avtorja:

Zahvala

Za pomoč pri diplomskem delu se zahvaljujem mentorici doc. dr. Mojci Ciglarič in celotni ekipi Laboratorija za računalniške komunikacije. Z njihovimi nasveti in konstruktivnimi kritikami so pripomogli k boljšemu končnemu izdelku diplomske naloge. Posebej bi se rad zahvalil tudi Tinetu Lesjaku, ki mi je na podlagi lastnih izkušenj pomagal premagati začetne ovire pri spoznavanju okolja VCL in asistentoma Matjažu in Andreju, ki sta preverila delovanje končnega izdelka diplomskega dela.

Staršem

Kazalo

Seznam uporabljenih kratic in simbolov

VCL (ang. Virtual Computing Lab) - Odprtokodna programska oprema, ki omogoča oskrbovanje z računalniški viri

EC2 (ang. Elastic Compute Cloud) - Amazonova spletna storitev, ki nudi razširljivo računalniško infrastrukturo v javnem oblaku

S3 (ang. Simple Storage Solution) - Amazonova rešitev za shranjevanje podatkov v medmrežju

AWS (ang. Amazon Web Services) - Amazonove spletne storitve

NCSU (ang. North Carolina State University) - državna univerza Severne Karoline v ZDA

API (ang. Application programming interface) - Programski vmesnik, ki zagotavlja, da ima računalniški program na razpolago funkcije drugega računalniškega programa

SSH (ang. Secure Shell) - varen protokol za upravljanje računalnika na daljavo

RDP (ang. Remote Desktop Protocol) - protokol za upravljanje računalnika z operacijskim sistemom Windows na daljavo.

HTTP (ang. Hyper Text Transfer Protocol) - Protokol za izmenjavo nadbesedil ter grafičnih, zvočnih in drugih večpredstavnostnih vsebin na spletu

REST (ang. Representational State Transfer) - Tip spletne arhitekture v porazdeljenem sistemu, kjer gre za komunikacijo med strežnikom in odjemalcem

SOAP (ang. Simple Object Access Protocol) - Protokol za izmenjavo strukturiranih informacij med spletnimi storitvami

SQL (ang. Structured Query Language) - Strukturiran povpraševalni jezik za delo s podatkovnimi bazami

XML (ang. Extensible Markup Language) - Razširljivi označevalni jezik, ki označuje format podatkov za izmenjavo strukturiranih dokumentov v spletu

WSDL (ang. Web Services Description Language) - Označevalni jezik za opis funkcionalnosti spletnih storitev

IaaS (ang. Infrastructure as a Service) - storitveni model oblaka, ki ponuja infrastrukturo

PaaS (ang. Platform as a Service) - storitveni model oblaka, ki ponuja platformo na kateri lahko izvajamo aplikacije

SaaS (ang. Software as a Service) - storitveni model oblaka, ki ponuja programsko opremo

xCAT (ang. Extreme Cloud Administration Toolkit) - zbirka orodij za upravljanje strežniških gruč

HPC (ang. High-Performance Computing) - visoko zmogljivo računalništvo

AMI (ang. Amazon Machine Image) - slika virtualnega okolja v EC2, zajema programski sklad operacijskega sistema in aplikacij

EBS (ang. Elastic Block Store) - Amazonova storitev, ki zagotavlja virtualne diskovne enote

Povzetek

V Laboratoriju za računalniške komunikacije trenutno uporabljamo okolje Virtual Computing Lab (VCL) za potrebe izvajanja vaj pri različnih predmetih. Sistem študentom omogoča dostop do virtualnih okolij, na katerih študentje izvajajo svoje naloge. Sestavljen je iz zasebnega oblaka 11 strežnikov, ki s pomočjo virtualizacijske tehnologije VMware omogočajo oskrbovanje z virtualnimi računalniki. Zasebni oblak je večino časa slabo izkoriščen, ob določenih obdobjih pa se pojavljajo povečane potrebe po virih, ki preobremenijo sistem. Za omenjeni problem v diplomskem delu predlagamo rešitev v obliki hibridnega oblaka. Obstoječ zasebni oblak bi normalno deloval še naprej, v primerih preobremenjenosti pa bi imeli možnost uporabe virov javnega oblaka. V ta namen smo razvili modul VCL za oskrbovanje z viri javnega oblaka Amazon EC2. S hibridnim oblakom rešimo problem preobremenjenosti obstoječega zasebnega oblaka in dosežemo boljšo izkoriščenost virov. Pri javnem oblaku plačujemo le dejansko porabo računskih ciklov in prostora, ki ga zasedamo. S tem dosežemo boljšo stroškovno učinkovitost v primerjavi z nakupom dodatne fizične strojne opreme oziroma nadgradnjo obstoječega zasebnega oblaka. Za razumevanje delovanja hibridnega oblaka smo v diplomskem delu predstavili področje računalništva v oblaku, opisali arhitekturo odprtokodnega sistema VCL in obravnavali delovanje javnega oblaka EC2. Za našo rešitev smo izbrali slednjega, saj je Amazon EC2 trenutno eden izmed vodilnih ponudnikov. Na koncu se seznanimo z načinom integracije preko vmesnika javnega oblaka in predlagamo arhitekturo hibridnega oblaka. Kot dokaz o uspešni integraciji hibridnega oblaka, njegovo delovanje prikažemo z razvijalčevega vidika in vidika končnega uporabnika.

Ključne besede:
računalništvo v oblaku, Virtual Computing Lab, Amazon EC2, oskrbovanje z viri, hibridni oblak

Abstract

Virtual Computing Lab (VCL) environment at Computer communications lab enables students remote access to virtual environments, in which they fulfill their course assignments. It is comprised of 11 physical servers with VMware hardware virtualization that allows provisioning of virtual machines. Most of the time the private cloud in underutilizated, while it gets overutilizated during usage peaks at the end of semester or around deadlines for certain student assignments. In this thesis we suggest a solution for the overutilization problem in form of a hybrid cloud. Current private cloud could still be used as it was untill now, but we would have a possibility to expand available resources with public cloud during usage peaks. To address this issue we have developed a new VCL provisioning module for provisioning Amazon EC2 public cloud resources. Hybrid cloud solution expands capabilites of VCL environment during usage peaks and provides optimal resource utilization. With public cloud's "pay-as-you-go" model we only pay for resources (compute cycles, storage) that we actually use. With hybrid cloud we achieve better cost efficency, than we would have with investments into new physical infrastructure to meet the demand for resources during usage peaks. To really understand how hybrid cloud solution works, we discuss cloud computing, describe open-source VCL environment and EC2 public cloud. In the end we explain how to integrate hybrid cloud with connecting via public cloud's API interface and suggest hybrid cloud's architecture in case of Computer communications lab. The result of this thesis is a working proof-of-concept hybrid cloud solution for VCL environment.

Key words:
cloud computing, Virtual Computing Lab, Amazon EC2, resource provisioning, hybrid cloud

Poglavje 1

Uvod

1.1 Motivacija

Laboratorij za računalniške komunikacije trenutno sestoji iz zasebnega oblaka, ki zaenkrat zadostuje potrebam za izvajanje laboratorijskih vaj. Oblak je sestavljen iz prilagojenega odprtokodnega sistema VCL (ang. Virtual Computing Lab), ki omogoča oskrbovanje z računalniški viri. Študentom in osebju omogoča rezerviranje virtualnih okolij z enim ali več virtualnimi računalniki, do katerih lahko dostopajo od doma oziroma s fakultete. Sistem poganja 11 fizičnih strežnikov, ki skupaj zagotavljajo približno 300 virtualnih računalnikov. Njegove kapacitete so omejene. Sistem je veliko časa slabo izkoriščen, predvsem med poletnimi meseci, ko so študijske počitnice. Preobremenjen pa je v obdobjih tik pred koncem semestrov in pred roki za oddajo seminarskih nalog. Prav tako se v času laboratorijski vaj povečajo zakasnitve sistema, ko množica študentov istočasno zažene večje število virtualnih računalnikov. V diplomskem delu poizkušamo najti rešitev za omenjene probleme. Ali fakulteta investira v nakup nove fizične strojne opreme in s tem zveča zmogljivost virtualnega laboratorija ter zmanjša izkoriščenost virov ali pa sistem poveže z javnim oblakom in plačuje le najemnino za uporabo virov. Odločili smo se za slednje in predlagali rešitev v obliki hibridnega oblaka. Obstoječ zasebni oblak bi normalno deloval še naprej, v primerih preobremenjenosti pa bi imeli možnost uporabe virov javnega oblaka.

Osnovna motivacija je bila kako zadostiti povečanim potrebam po virih ob konicah, ki se pojavljajo nekajkrat na leto. S povezavo z javnim oblakom pa se nam odpre še mnogo drugih možnosti, ki pridejo prav tudi v izobraževalnih ustanovah. Javni oblaki ponujajo navidezno neomejene računalniške zmogljivosti. Hibridni oblak omogoča raziskovalcem, da zahtevno procesiranje pre-

selijo v javni oblak in tako hitreje pridejo do rešitev. Fakulteta za reševanje nekaterih računsko zahtevnih nalog enostavno nima zadostnih zmogljivosti, oziroma bi bila investicija v drago strojno opremo nesmiselna, v kolikor ta ne bi bila polno izkoriščena. Javni oblaki se neprestano razvijajo in so začeli ponujati tudi visoko zmogljivostne računalnike ter virtualne oblake. Odveč je poudariti, da je taka oprema izjemno draga, raziskovalcu pa neprecenljivo, če mu fakulteta omogoči dostop do omenjenih virov, čeprav zgolj za omejen čas.

Prav tako se zdi smiselna uporaba javnega oblaka za varnostne kolokacije. Fakulteta bi lahko svojo infrastrukturo podvojila z virtualnimi viri z minimalnimi kapacitetami javnega oblaka, v primeru izpada lastne infrastrukture bi enostavno preusmerila promet v javni oblak in povečala zmogljivost virtualnih virov ter s tem zadostila dejanskim potrebam. Tako bi med obdobjem izpada lastne infrastrukture najeli več kapacitet, po vzpostavitvi sistema nazaj v prvotno stanje pa bi le-te zmanjšali na nivo, ki je potreben za varnostno preslikavanje.

Dodano vrednost hibridnega oblaka vidimo tudi v primeru laboratorijskih vaj, ki temeljijo na programski opremi določenega proizvajalca. Trenutno to poteka tako, da si študentje namestijo programsko opremo na svoje lastne računalnike oziroma uporabijo računalnike v laboratoriju, na katere je orodja namestil administrator. Namestitve so lahko dolgotrajne in mnogokrat študentje nimajo dovolj zmogljive strojne opreme za poganjanje zahtevnih programskih orodij. Veliko bolj enostavno bi bilo, da bi se fakulteta s proizvajalcem programske opreme (npr. IBM, Microsoft) dogovorila za uporabo njihovega javnega oblaka, kjer so okolja že vzpostavljena. Fakulteta ima s proizvajalci dogovor glede licenc, tako da bi lahko obstoječ dogovor le nadgradili z uporabo njihovih virtualnih virov iz javnega oblaka.

Ključna prednost vpeljave hibridnega oblaka v okolje VCL pa je v tem, da poteka oskrbovanje z viri javnega in zasebnega oblaka centralizirano preko enotnega sistema. Ne glede na to ali gre za študente, pedagoge, raziskovalce ali osebje za vzdrževanje računalniške infrastrukture. VCL omogoča sofisticiran način dodeljevanja pravic različnim uporabnikov, kar omogoča da raziskovalcem dodelimo pravico za uporabo visoko zmogljivih virtualnih računalnikov za neomejen čas, študentom pa omogočimo dostop do standardnih zmogljivosti in dodamo časovno omejitev uporabe. Prav tako imamo možnost, da fakulteta za vse porabljene vire od ponudnika javnega oblaka prejme enoten račun.

1.2 Zgradba diplomskega dela

Namen diplomskega dela je dokazati koncept uporabe hibridnega oblaka v izobraževalni ustanovi. Zato smo obstoječe okolje VCL v Laboratoriju za računalniške komunikacije razširili z javnim oblakom ponudnika Amazon. Razvili smo nov modul za oskrbovanje z viri javnega oblaka Amazon EC2 in preizkusili delovanje. Za razumevanje delovanja hibridnega oblaka je potrebno predstaviti široko področje računalništva v oblaku.

Na začetku bomo obravnavali temo računalništva v oblaku, ki je razmeroma novo področje in se neprestano razvija. Opisali bomo različne tipe oblakov in modele storitev v oblaku. Za podrobnejšo predstavitev bomo naredili primerjavo različnih modelov konkretnih ponudnikov javnega oblaka. Spoznali se bomo z nekaterimi nevarnostmi javnih oblakov in načini, kako se zavarovati pred njimi.

V 3. poglavju predstavimo okolje VCL. VCL je odprtokodni sistem za oskrbovanje z računalniškimi viri. Omogoča rezervacije fizičnih računalniških virov, virtualnih računalnikov in visoko zmogljivih računalnikov. Preko spletnega vmesnika končnim uporabnikom omogoča rezervacije virov in administracijo celotnega sistema. Sistem je modularen in prilagodljiv. Razvit je bil na državni univerzi Severne Karoline v ZDA (NCSU [`http://vcl.ncsu.edu/`]), leta 2008 pa so izvorno kodo predali odprtokodni skupnosti Apache Software Foundation (ASF). Predstavili bomo njegovo arhitekturo in opisali rešitve za shranjevanje podatkov, varnostne vidike sistema in primere uporabe.

Okolje VCL predstavlja zasebni oblak, namen diplomskega dela pa je razširitev v hibridni oblak. Kot najprimernejšega ponudnika javnega oblaka za integracijo hibridnega oblaka smo izbrali Amazon Elastic Compute Cloud (EC2). V 4. poglavju obravnavamo njegovo delovanje, orodja za upravljanje z viri javnega oblaka in na kratko predstavimo, katere tipe instanc ponuja. Posebno pozornosti posvetimo varnosti, saj je ta v javnem oblaku zelo pomembna.

Bralec je po 4. poglavju seznanjen s ključnima elementoma za integracijo hibridnega oblaka, zasebnim in javnim oblakom. Zato sledi poglavje, kjer podrobno predstavimo konkretno realizacijo hibridnega oblaka okolja VCL in oblaka EC2. Poglavje je bolj tehnične narave in opisuje predlagano arhitekturo hibridnega oblaka. Opišemo način, kako preko vmesnika API povežemo javni in zasebni oblak, predstavimo podrobnosti namestitve okolja VCL ter način gradnje modulov. Obravnavamo konkretno delovanje modula z razvijalčevega vidika in vidika končnega uporabnika. Na koncu kritično ocenimo našo rešitev in predstavimo prednosti in slabosti hibridnega oblaka.

Zaključke diplomskega dela predstavimo v zadnjem poglavju in predlagamo

smernice za nadaljnje delo s hibridnim oblakom.

Poglavje 2

Računalništvo v oblaku

Koncept računalništva v oblaku spreminja načine vzpostavljanja infrastrukture IT v podjetjih, izobraževalnih ustanovah, raziskovalnih zavodih in javni upravi. Zanimanje za tehnologijo se je zadnja leta povečevalo in na trgu so se pojavile različne rešitve. Oblak je dosegel široko skupino uporabnikov, od katerih se marsikdo sploh ne zaveda, da uporablja storitve računalništva v oblaku.

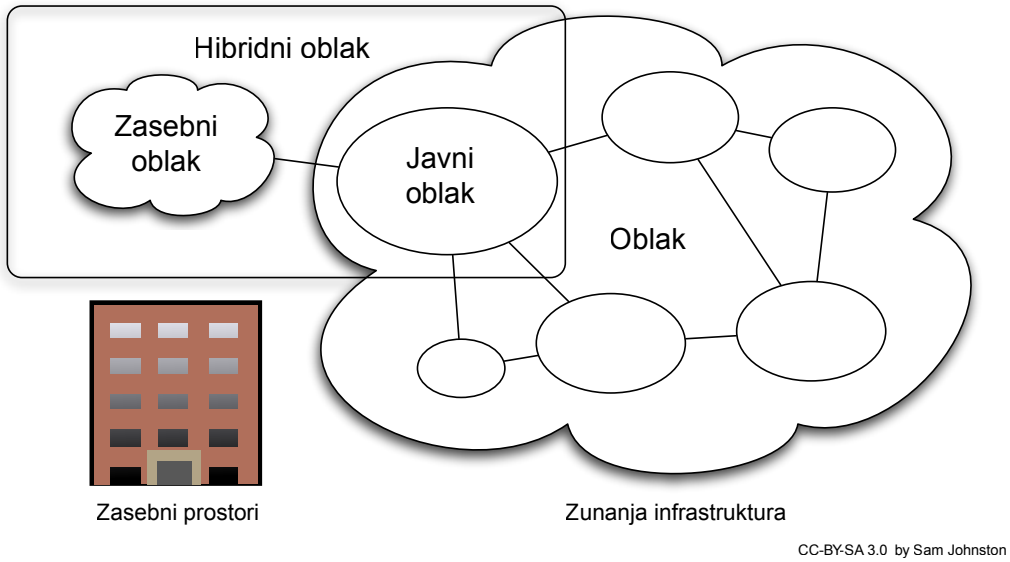
Definicija po ameriškem Nacionalnem inštitutu za standarde in tehnologijo [3]:

> *Računalništvo v oblaku je model, ki omogoča primeren omrežni dostop na zahtevo iz katerekoli lokacije do deljene množice nastavljivih računalniških virov (npr. omrežij, strežnikov, sistemov za shranjevanje podatkov, aplikacij in storitev), ki jih lahko hitro pridobimo in sprostimo z minimalnim upravljalnim trudom oziroma minimalno interakcijo ponudnika storitve. Model oblaka zagotavlja razpoložljivot virov in je sestavljen iz petih osnovnih karakteristik, treh storitvenih modelov in štirih namestitvenih modelov.*

Osnovne karakteristike so: dostop na zahtevo, omrežni dostop za širok razpon naprav, dinamično dodeljevanje virov, elastičnost in merljivost uporabe storitve. Namestitvene in storitvene modele oblaka obravnavamo v naslednjih podpoglavjih. Podobno kot naravni oblak se tudi sam tehnološki izraz spreminja tako hitro, da mu je včasih izjemno težko slediti. Vsekakor bo tehnologija pustila globoke posledice v industriji IT. Računalništvo v oblaku je termin, ki se je mnogo let počasi razvijal na hrbtih drugih tehnologij in je naravni rezultat evolucije teh tehnologij. Zanimanje za tehnologijo je vedno večje, prav tako pa se pojavlja vedno več uporabnikov storitev računalništva v oblaku.

2.1 Namestitveni modeli oblakov

Oblaki se med seboj razlikujejo glede na način namestitve. Poznamo zasebni, javni in hibridni oblak. Na sliki [2.1] so prikazana razmerja med posameznimi modeli. Zasebni oblak zajema infrastrukturo, ki jo koristi in upravlja določena organizacija. Ponavadi se vsi viri nahajajo na isti lokaciji. Javni oblak upravljajo ponudniki in svoje vire nudijo zainteresiranim strankam. Točna lokacija infrastrukture končnemu uporabniku ni znana. Hibridni oblak je sestavljen iz obeh, javnega in zasebega oblaka. Oblaka sta še vedno ločeni entiteti, vendar preko vmesnikov omogočata vzajemno delovanje.



Slika 2.1: Namestitveni modeli oblaka

2.2 Modeli storitev računalništva v oblaku

Osnovni modeli storitev računalništva v oblaku so:

- Infrastructure as a Service (IaaS) – dostop na zahtevo do uporabniško definiranih strojnih zmogljivosti, povezljivosti in pomnilniških kapacitet. Storitve se lahko izvajajo na poljubni fizični strojni opremi, ki končnemu uporabniku ni znana.

- Platform as a Service (PaaS) – dostop na zahtevo do uporabniško definiranega nabora virtualizacijskega okolja, operacijskega sistema in vmesnega sloja programske opreme (ang. middleware), ki zagotavlja delovanje uporabniških aplikacij in storitev. PaaS se lahko izvaja nad plastjo HaaS oziroma IaaS. Končni uporabnik dobi nadzor nad operacijskim sistemom.
- Software as a Service (SaaS) – končni uporabnik uporablja programsko opremo in shranjuje podatke, samo delovanje storitve pa ga ne zanima.

Pojavljajo se tudi izrazi:

- Hardware as a Service (HaaS) – dostop na zahtevo do točno določene strojne opreme na točno določeni lokaciji. Strojna oprema lahko zajema strežnike, sisteme za shranjevanje podatkov, mrežno opremo itd. Končni uporabnik se zaveda, kakšno fizično strojno opremo hoče uporabljati in pozna njeno lokacijo.
- Application as a Service (AaaS) – dostop na zahtevo do aplikacij.
- Cloud as a Service (CaaS), Security as a Service, Portabilitiy as a Service, Storage as a Service.

2.3 Primerjava modelov

Z virtualizacijo virov dosežemo elastičnost in iluzijo o neskončni zmogljivosti računalniških virov, sama implementacija deljenja in multipleksiranja virov pa je lahko skrita pred razvijalcem-uporabnikom. Različne ponudbe računalništva v oblaku se razlikujejo glede na stopnjo abstrakcije in načina upravljanja z viri [1].

Amazon EC2 je predstavnik prvega dela spektra. Ena instanca EC2 je z vidika uporabnika zelo podobna običajni strojni opremi. Nadzorujemo celoten programski sklad od jedra operacijskega sistema navzgor. Na instanci lahko gostimo katerikoli tip aplikacije. Nizek nivo virtualizacije omogoča razvijalcem, da uporabljajo instance na enak način, kot to počnejo z običajnimi fizičnimi strežniki. Na drugi strani pa to Amazonu onemogoča, da bi ponudili samodejno skalabilnost, saj je ta odvisna predvsem od aplikacijske plasti.

Amazon sicer ponuja tudi storitve na višjih nivojih abstrakcije, ki jih lahko uporabljamo v kombinaciji z EC2, vendar se te ne uporabljajo v taki meri, predvsem zaradi zakasnitev in nestandardnih vmesnikov API. Amazon EC2 je tipičen predstavnik modela IaaS.

Na drugi strani spektra storitvenih modelov računalništva v oblaku pa sta domensko specifični platformi Google AppEngine in Force.com. AppEngine je namenjen izključno tradicionalnim spletnim aplikacijam, ki morajo temljiti na "zahteva-odziv" (ang. request-response) arhitekturi. Te omejitve omogočajo AppEngine-u in sistemu za shranjevanje podatkov BigTable[1] samodejno skaliranje in visoko razpoložljivost. Prav zaradi teh omejitev pa AppEngine ni primeren za splošnonamenske aplikacije.

Microsoft Azure je nekje vmes med omenjenima modeloma. Azure aplikacije uporabljajo .NET knjižnice in se izvajajo v okolju Common Language Runtime. Sistem podpira splošnonamenske aplikacije in ne le neko specifično kategorijo. Uporabniki si lahko izberejo poljuben programski jezik (ki ga podpira CLR), ne morejo pa vplivati na operacijski sistem in izvajalno okolje. Knjižnice omogočajo določeno mero skalabilnosti, vendar mora razvijalec to definirati v aplikaciji. To naredi z opisom nekaterih lastnosti aplikacije. Azure je vmesna postaja med popolnoma aplikacijskim ogrodjem (AppEngine) in virtualnimi računalniki (Amazon EC2), saj vsebuje lastnosti obeh.

Kateri model je najboljši? Vprašanje je analogno vprašanju programskih jezikov. Nizkonivojski jeziki, kot so C in assembler, omogočajo zelo natančen nadzor, vendar za razvijanje spletnih aplikacij niso primerni. Na drugi strani visokonivojski jezik, na primer Ruby on Rails[2], zakrije podrobno delovanje in je uporaben le za spletne aplikacije. Za vsak odklon od običajne uporabe, bi bilo potrebno poseči v samo ogrodje, kar pa ni zaželjeno. Noben razvijalec Rubya ne bi ugovarjal hitrosti in zmogljivosti jezika C in obratno, razvijalec Cja ne bi oporekal Rubyu. Prav tako se bodo na področju računalništva v oblaku obdržali različni modeli glede na potrebe in povpraševanje.

Če nadaljujemo z analogijo s programskimi jeziki, prav tako kot lahko visokonivojske jezike implementiramo z nizkonivojskimi, bi lahko višjenivojske modele oblaka gostovali na nižjenivojskih. Na primer, AppEngine bi lahko gostoval na platformah Azure in EC2, Azure bi lahko gostoval na EC2. Vsekar pa ima vsak model svoje prednosti in slabosti. Končna odločitev, katerega izbrati pa je na strani uporabnika.

[1] BigTable - distribuirana podatkovna baza podjetja Google

[2] Ruby on Rails - je odprtokodna spletna platforma, ki omogoča hiter in enostaven razvoj spletnih aplikacij [`http://rubyonrails.si/`]

2.4 Nevarnosti javnega oblaka

Pri uporabi javnega oblaka smo glede zanesljivosti popolnoma odvisni od ponudnika storitev. V SLA (ang. Service Licence Agreement) so ponavadi zapisana stroga pravila glede delovanja in dostopnosti storitev, vseeno pa lahko pride do izpadov delovanja. Ostale nevarnosti javnega oblaka so še: podražitve cene storitev, vkleščenje podatkov v posamezen oblak (ang. data lock-in), slaba interoperabilnost med oblaki zaradi nestandardiziranih vmesnikov. Najbolj očitni pa so izpadi delovanja, v zadnjih časih smo bili večkrat priče spletnim mrkom, ki so bili posledica nedelovanja javnih oblakov. Vedno več podjetij za svoje storitve uporablja infrastrukturo javnega oblaka, zato so v javnosti taki izpadi delovanja zelo odmevni.

2.4.1 Izpadi delovanja

Amazon svoj javni oblak deli na med seboj neodvisne enote regije in znotraj njih manjša, bolj soodvisna razpoložljiva območja. V regiji ZDA vzhod je 21. aprila 2011 zaradi nepravilne izvedbe načrtovane nadgradnje izpadel eden izmed nosilcev podatkov, ki ni mogel izvajati zahtev po pisanju ali branju podatkov [15]. Pred nadgradnjo sistema bi bili morali preusmeriti promet s pomožnih usmerjevalnikov na glavno omrežje, a so pomotoma storili ravno nasprotno. Pomožno omrežje obremenitve ni zdržalo in prišlo je do izpada delovanja. Napaka se je potem kaskadno razširila znotraj regije še na ostala razpoložljiva območja, saj so delujoči podatkovni nosilci poizkušali sinhronizirati svojo vsebino na nove lokacije (postopek se imenuje preslikavanje in se izvede, ko primarna varnostna kopija postane nedosegljiva), kar je preobremenilo celotno omrežje. Napaka je povzročila nedostopnost 13 odstotkov zapisov v regiji ZDA vzhod.

Amazon se je opravičil vsem prizadetim stranem in jim ponudili kompenzacijo v višini zneska, ki ga plačujejo za 10-dnevni najem Amazonovih storitev. Hkrati so sprejeli ukrepe za izboljšanje omrežja in preprečitev podobnih težav v prihodnosti. Predvsem bodo strankam olajšali delo z več razpoložljivimi območji in regijami, tako da bodo imele večjo redundanco. Izpad je povzročil človeški faktor. Čeprav je šlo za kombinacijo nesrečnih naključij in slabe konfiguracije omrežja, je celotno kaskado sprožila napačna poteza pri nadgradnji sistema.

2.4.2 Zaščita pred izpadi

Podjetje Rightscale ponuja platforme za upravljanje aplikacij pri različnih ponudnikih računalništva v oblaku. Ob nedavnem izpadu Amazonovega EC2 oblaka so zaznali naslednje ugotovitve in jih zapisali v spletnem dnevniku [10].

Varnostne kopije in repliciranje podatkov je potrebno vzeti resno. V primeru EC2 to pomeni repliciranje preko različnih razpoložljivh območij. Prav tako je treba v primeru izpada povečati zmogljivosti replik, da prenesejo obremenitve ob izpadih glavnih strežnikov. Primer dobre prakse: aplikacijo namestimo na strežniške instance na treh razpoložljivih območjih in nastavimo samodejno skaliranje na 30-60% izkoriščenosti virov. S tem dosežemo dovolj zmogljivosti za obremenitvene konice. Če odpove eno razpoložljivo območje, se zviša izkoriščenost virov na 90%. Verjetnost, da odpove več razpoložljivih območij hkrati, je majhna, a vseeno ne nična. Izpad delovanja storitev shranjevanja podatkov (AWS S3) se je zgodil 20. junija 2008, ko storitev ni delovala celo na različnih regijah (ZDA in Evropa). Veliko lahko dosežemo z repliciranjem podatkov v oblaku, vseeno pa je še vedno priporočeno imeti varnostne kopije na lastni infrastrukturi.

Poglavje 3

VCL - Virtual Computing Lab

3.1 Ozadje VCL

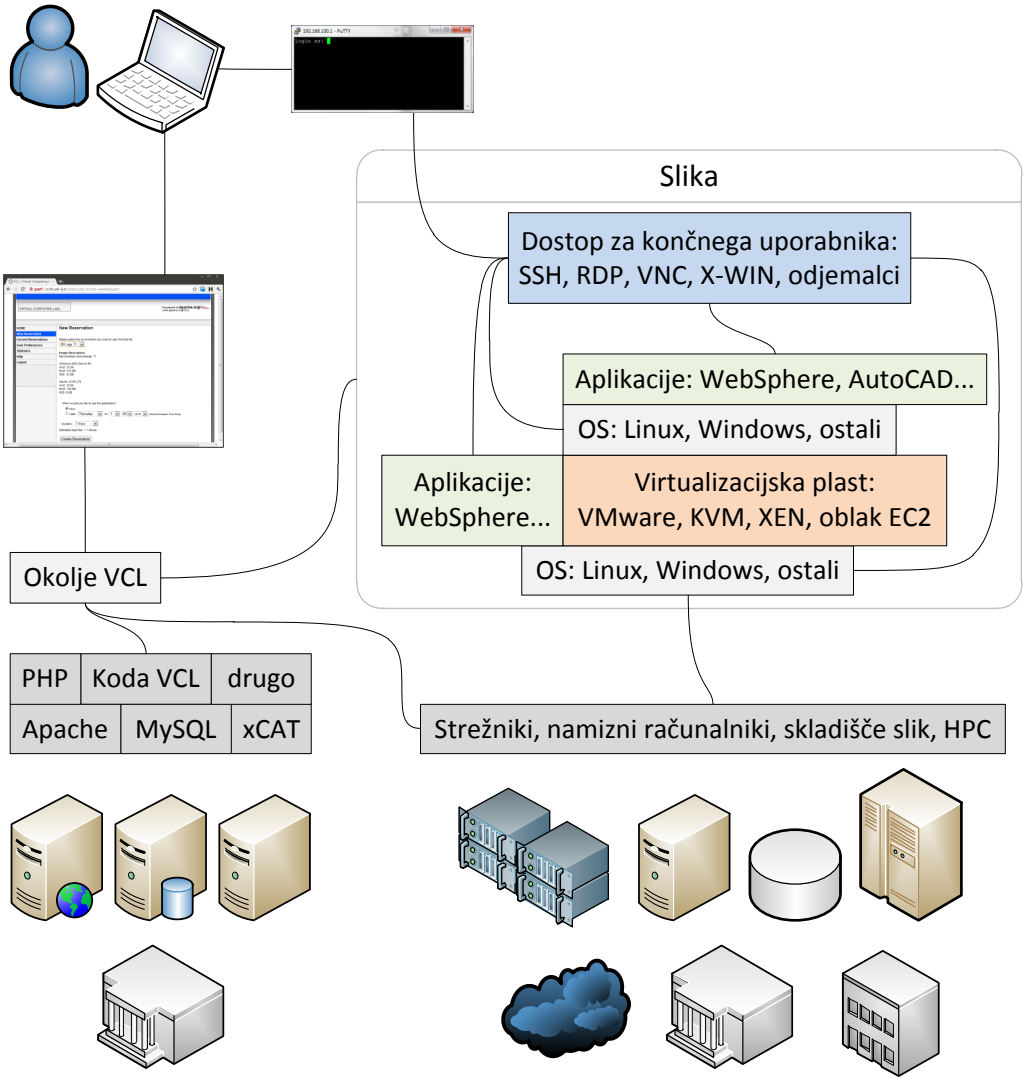
Projekt VCL je leta 2004 zrasel iz preproste ideje, kako zagotoviti dostop do množice računalniških okolij za študente in raziskovalce iz katerekoli oddaljene lokacije povezane v omrežje. Na državni univerzi Severne Karoline v ZDA (NCSU) so hoteli izboljšati izkoriščenost svojih virov IT in se lotili razvoja sistema za upravljanje z računalniškimi viri. Uspešno so razvili sistem, ki je temeljil predvsem na oskrbovanju s fizičnimi računalniki. Z razvojem virtualizacije in računalništva v oblaku, pa so sistem še razširili z novimi tehnologijami. Univerza je leta 2008 izvorno kodo okolja VCL predala odprtokodni skupnosti Apache Software Foundation (ASF) in s tem razširila krog uporabnikov VCL ter pospešila razvoj. Sama še vedno sodeluje pri razvoju sistema in ga tudi aktivno uporablja. Naslednja poglavja so povzeta po *[4, 5, 6]*.

3.2 Visokonivojska arhitektura VCL

Visokonivojska arhitektura VCL je v osnovi namenjena načrtovanju in nameščanju oblaka, ki bo služil izobraževalnim in raziskovalnim namenom univerz na čim bolj ekonomičen način. VCL ponuja veliko funkcionalnosti in storitev, ki sovpadajo s pričakovanji in zahtevami računalništva v oblaku. Glavne komponente VCL arhitekture so prikazane na sliki [3.1].

- spletni grafični vmesnik za končnega uporabnika
- podatkovna baza
- upravljalec z viri (upravljalec VCL)

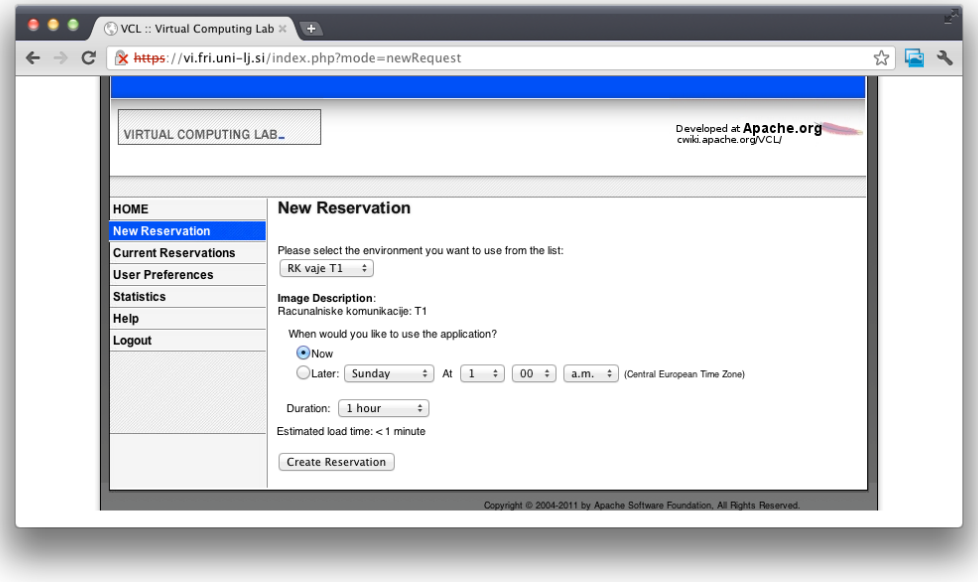
- slike: programski sklad, ki zajema operacijski sistem in prednameščeno programsko opremo, slike so shranjene v skladišču slik
- strojna oprema: fizični strežniki, rešitve za shranjenje podatkov, omrežna oprema, visoko zmogljivi računalniki, namizni računalniki
- varnost



Slika 3.1: Visokonivojska arhitektura VCL

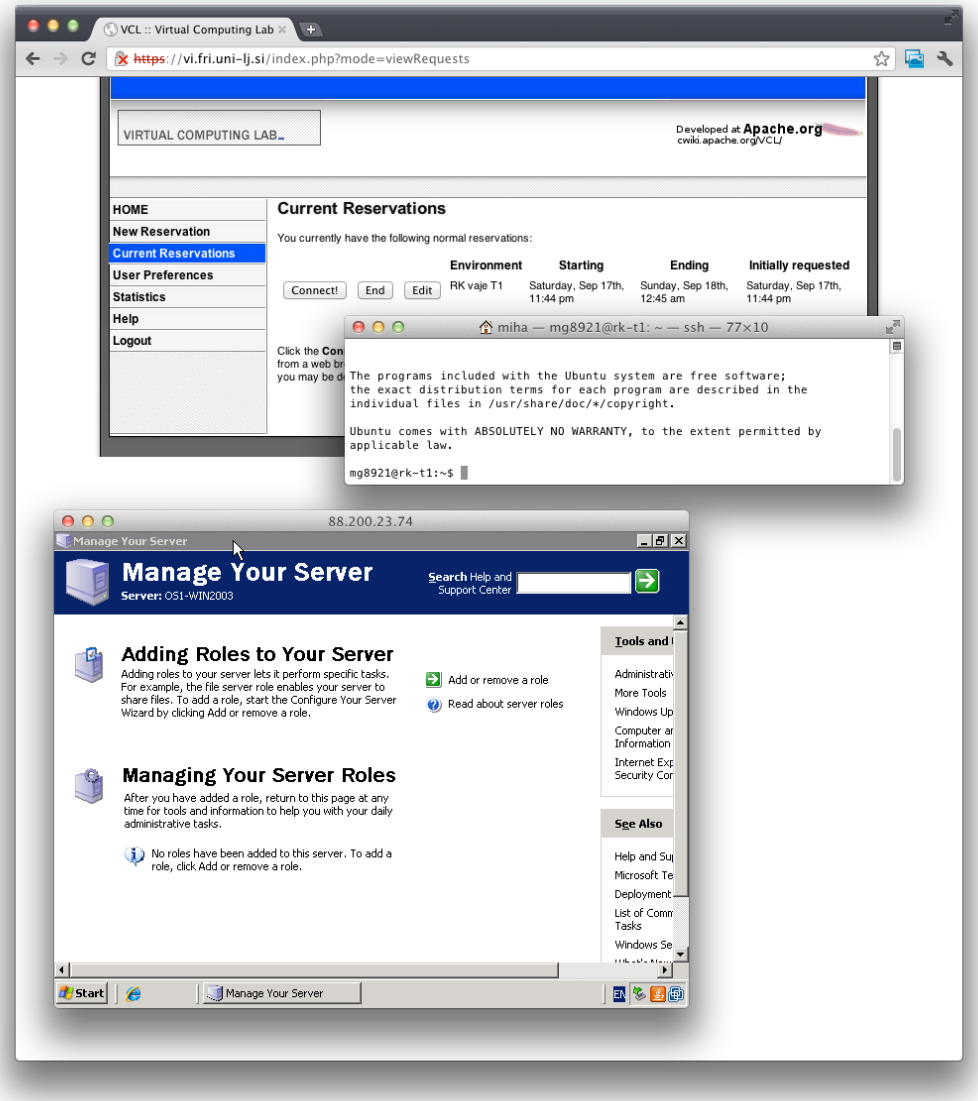
3.3 Uporabnik

Prvi stik uporabnika z VCL je preko spletnega vmesnika, kjer si izbere željeno kombinacijo programske opreme, ki jo želi uporabljati. Vmesnik je prikazan na sliki [3.2]. V primeru, da uporabniku ne ustreza nobena od ponujenih slik, si jo lahko prilagodi, če ima ustrezne pravice. Izbira lahko iz obstoječih komponent, ki jih ponuja VCL, in sliko povsem prilagodi svojim potrebam ter jo shrani za ponovno uporabo. Rezervira okolje z določenim operacijskim sistemom, nanj naloži svoje aplikacije in novo sliko shrani. Naslednjič si izbere prilagojeno sliko in mu ni potrebno ponovno nameščati programske opreme. Ko uporabnik izbere in rezervira sliko, upravljalec VCL obdela uporabniško zahtevo in sliko naloži na ustrezne strojne vire ter rezervirano okolje pripravi za takojšno uporabo oziroma rezervira za kasnejšo.



Slika 3.2: Nova rezervacija

Način dostopa do rezerviranih virov je odvisen od tipa slike, ki smo jo izbrali. Za okolja Windows se uporablja protokol RDP oziroma VNC, za okolja Linux pa protokol SSH oziroma X-WIN, ki omogočata dostop do oddaljenega strežnika. Uporabnik ob uspešni rezervaciji prejme dostopne podatke in lahko začne uporabljati rezervirano okolje. Na sliki [3.3] je prikazan seznam trenutnih rezervacij ter povezava do okolja Linux preko protokola SSH in povezava do okolja Windows preko protokola RDP.



Slika 3.3: Trenutne rezervacije in povezava do rezerviranih virov

3.4 Nizkonivojska arhitektura VCL

Tipično delovanje okolja VCL vključuje preverjanje aktivnih okolij, upravljanje z virtualnimi in fizičnimi računalniki ter upravljanje s slikami. Prav tako skrbi za interakcijo s končnimi uporabniki in vodi zgodovino rezervacij. Nizkonivojska arhitektura sestoji iz komponent, ki so predstavljene v naslednjih

podpoglavjih.

3.4.1 Okolja za oskrbovanje s fizičnimi ali virtualnimi viri

VCL lahko povežemo z okoljem za oskrbovanje s fizičnimi ali virtualnimi viri. Za fizične vire je v VCL previdena uporaba odprtokodne programske opreme Extreme Cloud Administration Toolkit (xCAT). Okolje xCAT je zbirka skriptnih orodij za izdelavo, nastavljanje, upravljanje in vzdrževanje strežniških gruč [14]. VCL uporablja xCAT za nalaganje slik na fizične strežniške rezine. Za oskrbovanje z virtualnimi viri je predvidena uporaba virtualizacijskih tehnologij VMware, možno pa je uprabiti tudi druge (npr. Oracle VirtualBox ali KVM). Za druge tehnologije je seveda potreben dodaten razvoj modulov, česar smo se lotili v diplomskem delu in omogočili oskrbovanje z virtualnimi viri javnega oblaka.

V samem začetku razvoja VCL (2004) je bil sistem pretežno usmerjen predvsem na fizične računalnike, saj se je virtualizacija računalniških virov šele začela razvijati. Danes VCL omogoča tako nalaganje slik na fizične računalnike, kot tudi na virtualne računalnike. Pomembno se je zavedati, da imajo v nekaterih primerih fizični računalniki še vedno nekaj prednosti pred virtualnimi, zato VCL tudi omogoča oboje. Predvsem na področju visoko zmogljivih računalnikov se opazijo razlike, ki so v prid fizičnimi virom.

VCL na podlagi uporabniške zahteve sproži postopek zagona določene slike. V primeru, da ne najde razpoložljivega fizičnega ali virtualnega strežnika, ki bi že imel prednaloženo zahtevano sliko, sam izbere najprimernejšega prostega kandidata, ki ustreza zahtevam obravnavane slike. Dinamično naloži sliko preko xCAT oziroma sistema za virtualizacijo virov. Na državni univerzi Severne Karoline v ZDA (NCSU) fizične slike nalagajo preko xCAT orodja, virtualne pa pa preko virtualizacijskih tehnologij VMware, predvsem sistema ESXi. V Laboratoriju za računalniške komunikacije prav tako uporabljamo VMware ESXi tehnologijo za virtualne računalnike, oskrbovanja s fizičnimi viri pa nismo implementirali, saj ni bilo potrebe.

V primeru, da so vsi viri zasedeni in ni mogoče naložiti zahtevane slike, VCL preko spletnega vmesnika uporabniku predlaga alternativne časovne termine, v katerih so ustrezni viri prosti.

3.4.2 Upravljalec VCL

Jedro upravljalca VCL je demonska storitev VCLd, ki skrbi za oskrbovanje virov in nalaganje slik. Storitev je napisana v programskem jeziku Perl. Pri praktičnem delu tega diplomskega dela smo se osredotočili predvsem na delovanje omenjene storitve, zato dobro poznavanje jezika Perl in njegovih knjižnic ni bilo odveč. Glede na zahtevano okolje, bodisi gre za fizično sliko bodisi za virtualno, VCLd zagotavlja, da se slika ustrezno naloži in omogoči dostop do nje.

Naloge storitve VCLd so:

- Komunikacija s spletnim vmesnikom in podatkovno bazo za pridobitev podatkov o rezervacijah virov, ki jih uporabniki vnesejo preko spletnega vmesnika. Ti so osnova za procesiranje rezervacij in upravljanje s slikami.
- Zaganjanje ukazov xCAT in VMware za izvrševanje uporabniških zahtev. Gre predvsem za nalaganje slik, zaganjanje virtualnih in fizičnih računalnikov, ugašanje računalnikov, ki jih ne potrebujemo itd.
- Nadziranje postopkov zajemanja novih slik, na katere namestimo novo programsko opremo.
- Vzdrževanje sistema za oskrbovanje z viri.
- Nastavljanje in administracija nameščenih slik glede na zahtevano uporabo.

3.4.3 Odprtokodni spletni strežnik Apache

Spletna aplikacija je napisana v programskem jeziku PHP in nameščena na spletni strežnik Apache. Predstavlja srce VCL in zagotavlja orodja za zahtevanje, upravljanje in obvladovanje vseh virov VCL. Spletni vmesnik skrbi za avtentikacijo uporabnikov, prikaže seznam vseh aplikacij, za katere je posamezen uporabnik avtoriziran in mu omogoči rezervacijo računalniških okolij bodisi za takojšnjo bodisi za kasnejšo rabo. Razpon časovnega obdobja (kdaj lahko rezerviramo vire ter samo trajanje rezervacije) je popolnoma prilagodljiv in se lahko razlikuje glede na uporabnikove pravice. Glavne funkcionalnosti spletnega vmesnika so:

- Ustvarjanje slik: vmesnik uporabniku omogoča ustvarjanje prilagojenih slik. Za osnovo uporabi obstoječe osnovne slike.

- Nadzor verzij slik: vmesnik uporabniku z ustreznimi pravicami omogoča ustvarjanje različnih verzij posamezne slike.
- Upravljanje z uporabniki: omogoča nastavljanje različnih nivojev pravic posameznim skupinam uporabnikov.
- Upravljanje z viri: omogoča način časovnega razporejanja vseh virov, ki so na voljo. Virom lahko določimo urnike s časovnimi obdobji, v katerih so na voljo za uporabo.

3.4.4 Odprtokodna podatkovna baza MySQL

Podatkovna baza hrani stanja vsakega računalniškega vira, vzdržuje metapodatke o slikah in virih, implementira sistem uporabniških pravic. Vse rezervacije se beležijo in imamo natančen pregled na dogajanjem v sistemu VCL. Na podlagi podatkov lahko analiziramo uporabo, kar nam omogočajo orodja znotraj spletnega vmesnika. Podatkovni model je dobro zasnovan in razširljiv, tako da omogoča tudi nadaljni razvoj funkcionalnosti, kot je na primer zaračunavanje oziroma vodenje uporabe virov po posameznih uporabnikih. Podatkovna baza temelji na odprtokodnem sistemu za upravljanje s podatkovnimi bazami MySQL [12], ki zagotavlja ustrezno okolje za transakcijsko intenzivno delovanje sistema VCL. Za dostop do baze VCL uporablja module, ki omogočajo enostavno zamenjavo sistema za upravljanje s podatkovnimi bazami in razvijalcu nudijo učinkovit dostop do podatkov. Razvijalcu se ni potrebno ukvarjati s podrobnostmi dostopanja do baze, za to poskrbijo metode modulov.

3.4.5 Slike

VCL sliko definira kot programski sklad, ki vsebuje naslednje plasti:

- Osnovni operacijski sistem
- Nameščene aplikacije in predvnešeni podatki
- Rešitev za dostop končnega uporabnika, ki ustreza operacijskemu sistemu (protokol SSH za Linux, protokol RDP za Windows)

Slike lahko nalagamo na fizične računalnike ali na katerokoli virtualno okolje. Na NCSU in FRI se za virtualizacijsko okolje uporablja VMware ESXi, lahko pa bi uporabili tudi drugo, na primer Oracle VirtualBox. Namen diplomskega

dela pa je poganjati slike v javnem oblaku, in sicer v Amazon EC2 virtualizacijskem okolju. Uporabnik z ustreznimi pravicami, si programski sklad lahko poljubno prilagodi. Kot osnovo vzame sliko z nameščenim osnovnim operacijskim sistemom (Linux ali Windows), poljubno spremeni sistemske nastavitve in naloži svoje aplikacije. Sliko shrani in omogoči dostop skupini uporabnikov, katerim je namenjena.

3.5 Strojna oprema v VCL

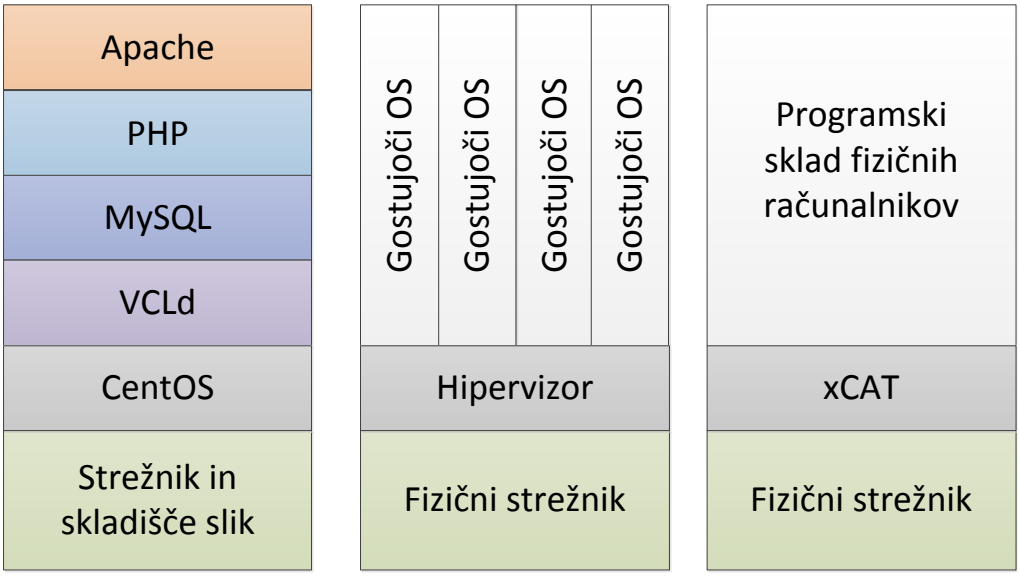
Virtualizacija pripelje abstrakcijo stojne opreme do te mere, da lahko programski sklad poljubno nalagamo na virtualne računalnike, ne da bi bili vezani na točno določeno strojno opremo. Strežniki VCL zagotavljajo široko množico virov, ki jih lahko ponudijo na zahtevo uporabnika. Viri se dodelijo glede na aplikacijske zahteve, računsko intenzivne slike dobijo več procesorske moči in večje pomnilniške kapacitete, slike, ki jih uporabljamo samo za testiranje komunikacij med računalniki, pa take moči ne potrebujejo in prejmejo ustrezno manjše zmogljivosti. Omrežni viri in sistemi za shranjevanje podatkov morajo biti razširljivi ter ustrezati obremenitvam in uporabniških zahtevam. V primeru pomanjkanja kapacitet oziroma zmogljivosti jih enostavno nadgradimo.

Zasebni oblak ponavadi dopolnjuje še sistem za shranjevanje podatkov, ki je na voljo vsem virtualnim računalnikom. VCL omogoča, da lahko vse vire uporabimo tako za izvajanje aplikacij kot tudi za shranjevanje podatkov. Pri shranjevanju podatkov pa moramo paziti, da je okolje nastavljeno na trajni način. Strojne vire lahko predstavljajo strežniške rezine, skupine namiznih računalnikov ali delovnih postaj, visoko zmogljivi računalniki itd.

Tipična namestitev VCL sestoji iz ene ali več strežniških šasij, ponavadi je ena strežniška rezina posvečena upravljalnemu vozlišču. Na NCSU uporabljajo eno upravljano vozlišče za oskrbovanje s približno 100 računalniki. Vsaka rezina mora imeti najmanj dva mrežna vmesnika – enega za povezavo z javnim omrežjem in drugega za zasebno omrežje, ki omogoča upravljanje z lokalnimi računalniškimi viri in nalaganje slik. Sistemi za shranjevanje podatkov so priključeni s pomočjo optičnih vlaken oziroma mrežnih povezav. Struktura tipične namestitve VCL je prikazana na sliki [3.4].

3.6 Varnost v VCL

Težko je podati definicijo, kaj pomeni varnost v oblaku. Glavno merilo pri porazdeljenih sistemih je urejen postopek avtentikacije in avtorizacije posa-



Slika 3.4: Struktura tipične namestitve VCL

meznih storitev v oblaku. VCL v osnovi implementira naslednja varnostna mehanizma:

Avtentikacija LDAP: avtentikacija VCL temelji na storitvi LDAP. Za različne uporabniške skupine omogoča različne storitve LDAP za ločen uporabniški dostop.

Avtentikacija na nivoju okolja: tak način avtentikacije je odvisen od okolja do katerega dostopamo. Ponavadi je določen ob zagonu slike. V okolju Windows privzeto uporabimo geslo za enkratno uporabo, ki se ustvari ob začetku rezervacije in poteče, ko končamo z uporabo. V okolju Linux lahko vzpostavimo obstoječ sistem avtentikacije ali pa prav tako ustvarimo enkratna gesla za točno določeno rezervacijo. Na NCSU praviloma za okolja Windows uporabljajo enkratna gesla, za okolja Linux pa obstoječ centraliziran sistem za avtentikacijo, ki omogoča dostop do vseh virov univerze z enotnim uporabniškim imenom in geslom. Razlog za to pa je predvsem v tem, da okolju Linux bolj zaupajo kot okolju Windows.

Poleg omenjenih mehanizmov avtentikacije lahko sistem dodatno zavarujemo z omejitvami na požarnem zidu. VCL omogoča preprečevanje dostopa

do rezerviranih virov glede na izvorni naslov IP. Uporabniku lahko omogočimo dostop do rezerviranih virov le prek IP naslova, preko katerega je tudi rezerviral vire. Tako lahko vire uporablja le iz lokacije, preko katere je oddal zahtevek za rezervacijo in s tem izboljšamo varnost sistema.

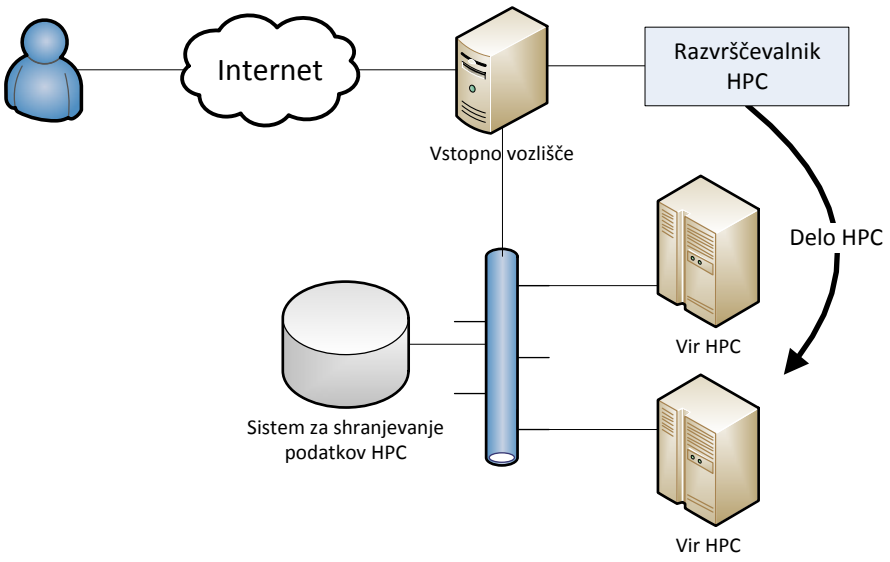
3.7 Visoko zmogljivo računalništvo in VCL

Na univerzah se vedno bolj pojavlja potreba po visoko zmogljivem računalništvu (ang. High-Performance Computing - HPC). Potrebuje se za reševanje računsko zelo zahtevnih problemov, kjer so v praksi fizični računalniki še vedno bolj učinkoviti od virtualnih. VCL je bil sprva načrtovan za oskrbovanje s fizičnimi viri, ta funkcionalnost pride prav ravno v primeru virov HPC, ki so seveda fizični. Osnovni model delovanja HPC in VCL je razmeroma enostaven. Prikazan je na sliki [3.5]. Uporabnik preko vstopnega vozilšča dostopa do virov HPC, s katerimi upravlja VCL. Z orodjem xCAT naloži slike na proste strežniške rezine, namenjene HPC, prav tako v omrežnih nastavitvah omogoči povezljivost rezin znotraj virtualnega zasebnega omrežja (VLAN). V okolju HPC je za varnost še bolje poskrbljeno in se za dostop preko javnih omrežij uporablja temu namenjena vstopna vozlišča. VCL vzpostavi okolje HPC vključno s komponentami, kot sta na primer izenačevalnik obremenitve in razvrščevalnik HPC. Vsaka HPC slika ima dostop do velike količine prostora v sistemih za shranjevanje (v TB), kot tudi do lastnih prostorskih kapacitet in sistemov za varnostne kopije. Ko se slika HPC naloži, razvrščevalnik VCL to zazna in ji začne predajati delo. Integracija HPC in VCL močno poveča izkoriščenost virov z izboljšanjem dodeljevanja rezin HPC. Tak način omogoča skupno rabo infrastrukture med več uporabniki, kar vodi v boljšo razpoložljivost in izkoriščenost virov. HPC in VCL sta podrobneje obravnavana v [7].

3.8 Prednosti uporabe VCL v laboratorijih

Našteli smo nekatere glavne zmogljivosti okolja VCL. Na univerzah so tipični uporabniki študentje, raziskovalci in pedagogi. Oblak za tako vrsto uporabnikov mora zagotavljati vsaj naslednje zmogljivosti:

- storitve in podporo za širok razpon uporabnikov
- veliko podpornih orodij za pedagoge in univerzitetno osebje
- primerne računalniške sisteme in storitve, ki služijo raziskovalni dejavnosti



Slika 3.5: Osnovni model delovanja HPC in VCL

Poleg upoštevanja omenjenih zahtev so glavni izzivi pri planiranju oblaka za izobraževalne in raziskovalne namene naslednji:

- čimboljša izkoriščenost virov glede na uporabniške zahteve
- raznovrstna storitvena okolja
- vzdrževanje in upravljanje infrastrukture mora bo ekonomično in stroškovno učinkovito

Na univerzah se potrebe po virih spreminjajo glede na študijski koledar. Ob koncih semestrov se zahteve po virih močno povečajo, prav tako narastejo ob rokih za oddajo različnih nalog, ki jih morajo opraviti študentje. Raziskovalno usmerjena dela potekajo skozi celo leto. Da zagotovimo stroškovno učinkovitost univerzitetnega oblaka, moramo vzpostaviti dober mehanizem razporejanja virov, ki spremlja zahteve in se jim prilagaja. VCL omogoča dobro razporejanje in prepoznavanje konic v uporabi med različnimi tipi uporabnikov.

Z opazovanjem okolja VCL na NCSU smo ugotovili pomembno prednost sistema, ki poleg običajnih strežnikov in namiznih računalnikov omogoča še izkoriščanje infrastrukture HPC. Tako VCL omogoča učinkovito izkoriščanje vseh razpoložljivih virov na NCSU. Z deljeno uporabo vseh strežniških rezin,

namiznih računalnikov in delovnih postaj ter virtualnih okolij ponuja ekonomično rabo virov in njihovo optimalno zasedenost.

S tem lahko zaključimo, da je VCL odprtokodni spletni sistem za dinamično oskrbovanje z viri in dodeljevanje oddaljenega dostopa do računalniških okolij na zahtevo uporabnika. Oblak VCL omogoča izjemno računsko moč s pomočjo odprte programske opreme in primerne strojne opreme, ki služi vsem univerzitetnim projektom in izobraževalnim programom.

Poglavje 4

Javni oblak Amazon EC2

4.1 Kaj je EC2

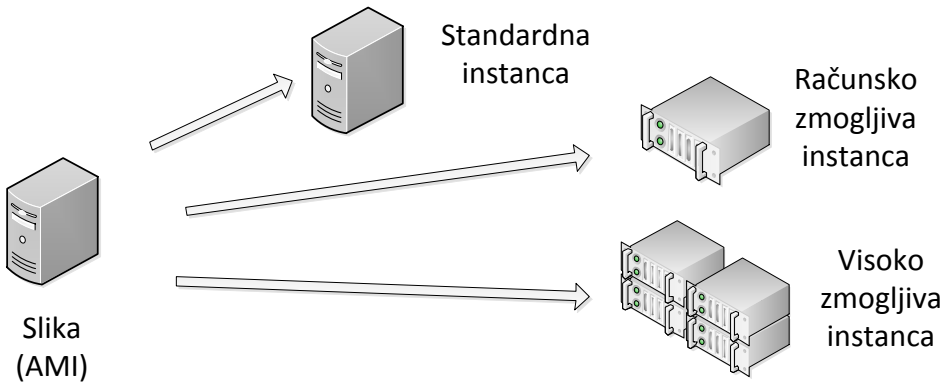
Amazon Elastic Compute Cloud (Amazon EC2) je spletna storitev, ki nudi razširljivo računalniško infrastrukturo. EC2 je predstavnik storitvenega modela IaaS. Javni oblak zagotavlja strežniške instance, na katerih lahko gradimo in gostujemo svoje aplikacije. Infrastrukturne vire EC2 upravljamo s pomočjo programskega vmesnika API, spletnega grafičnega vmesnika oziroma raznih orodij.

Pri EC2 uporabljamo in plačujemo le za zmogljivosti, ki jih dejansko uporabimo. To izniči potrebo po nakupih drage in prostorsko potratne strojne opreme, zmanjša potrebo po napovedovanju prometa in omogoča avtomatsko skaliranje virov IT in s tem prilagajanje konicam v prometu. Podrobno delovanje in načini uporabe javnega oblaka EC2 so opisani v Amazonovi dokumentaciji [9], glavni deli so povzeti v tem poglavju.

4.2 Osnovne komponente

4.2.1 Amazon Machine Images in instance

Amazon Machine Image (AMI) je slika, ki vsebuje operacijski sistem in prednameščene aplikacije. Iz slike lahko zaženemo instance, ki so aktivna virtualna okolja. Iz slike lahko zaženemo več instanc, kot je razvidno v sliki [4.1]. Instance so aktivne, dokler jih ne ustavimo oziroma odpovedo zaradi napake. Če posamezna instanca odpove, lahko iz predloge zaženemo novo. V oblaku lahko shranimo poljubno število različnih slik. Slika ni omejena s tipom instance. Tip instance predstavlja kombinacijo virtualne strojne opreme, ki je določena



Slika 4.1: Slika na različnih tipih instanc

s količino pomnilnika, računsko močjo in diskovnim prostorom. Podrobna tabela s tipi instanc in njihovimi zmogljivostmi je na voljo v dokumentaciji EC2 [9]. Isto sliko AMI lahko naložimo na katerikoli tip instance, odvisno od zahtev programske opreme, ki jo želimo izvajati na njej.

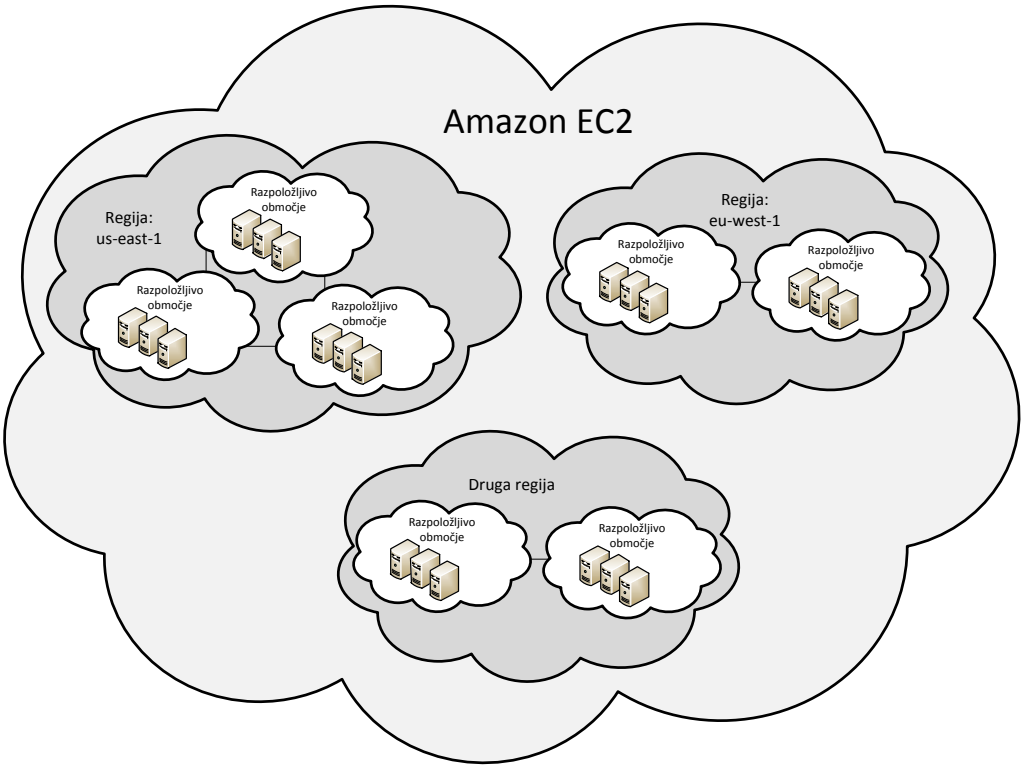
Amazon ponuja mnogo prednastavljenih slik AMI z najbolj pogostimi programskim nastavitvami. Predloge so proste dostopne in na voljo takoj. Dodatne slike ponuja tudi odprtokodna skupnost, ki redno objavlja najnovejše verzije operacijskih sistemov in programske opreme. Uporabljamo lahko prosto dostopne predloge in ustvarimo skripte, ki ob zagonu instance nastavijo sistem po naših željah. Lahko pa ustvarimo povsem svoje prilagojene slike AMI, ki zajemajo vse plasti programske sklada od operacijskega sistema navzgor. Svoje slike lahko prav tako delimo z ostalimi uporabniki oblaka EC2.

4.2.2 Regije in razpoložljiva območja

Amazon ima svoje podatkovne centre v različnih regijah sveta (Severna Amerika, Evropa, Azija itd.). Uporabnik lahko sam izbere, v kateri regiji bo uporabljal instance in se tako čim bolj približal končni stranki, ki bo uporabljala njegove storitve oziroma zadostil zakonskim predpisom, ki veljajo v nekaterih državah. Cena najema instanc se razlikuje glede na regije.

Vsaka regija vsebuje več različnih razpoložljivih območij. Vsako območje je zasnovano tako, da je izolirano pred odpovedmi drugih območij znotraj regije, vseeno pa zagotavlja poceni in hitro medsebojno omrežno povezavo z minimalnimi zakasnitvami. Z uporabo različnih razpoložljivih območij znotraj

regij, lahko zaščitimo svojo aplikacijo pred odpovedmi določenih lokacij in tako zagotovimo neprestano razpoložljivost svojih storitev. Za lažje razumevanje so regije in razpoložljiva območja prikazana na sliki [4.2].

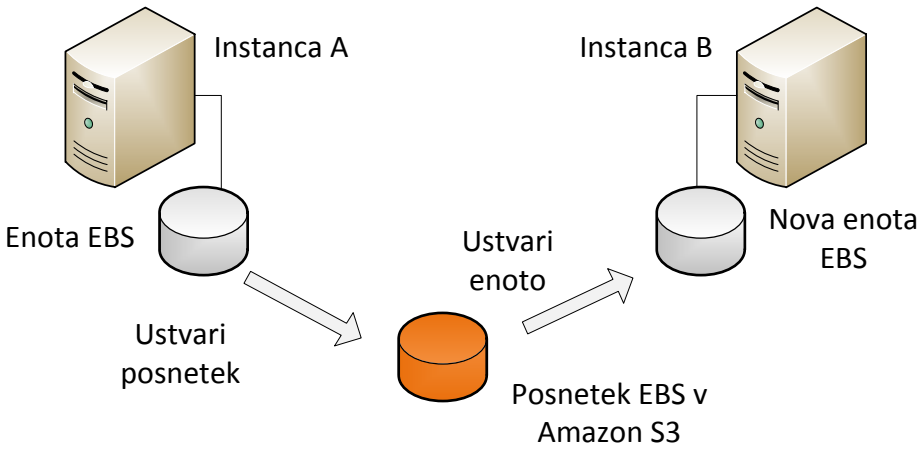


Slika 4.2: Regije in razpoložljiva območja Amazon EC2

4.3 Shranjevanje podatkov

Virtualni disk, ki pripada aktivni instanci, v osnovi ne zagotavlja trajnega shranjevanja podatkov in moramo za to poskrbeti sami. Podatki obstajajo, dokler ne ugasnemo instance. Zato moramo v teh primerih najprej shraniti podatke s pomočjo naslednjih storitev AWS:

Amazon S3 Amazon S3 (Simple Storage Solution) je storitev shranjevanja podatkov za celotno medmrežje. Preko enostavnega spletnega vmesnika



Slika 4.3: Posnetek EBS

lahko shranjujemo in pregledujemo kakršnekoli količine podatkov iz katerekoli lokacije v medmrežju.

Amazon EBS Amazon EBS je novejša storitev in zagotavlja trajno bločno shranjevanje podatkov. Enote EBS so virtualni diski, ki jih lahko priklopimo na aktivne instance. Enote so posebej primerne za aplikacije, ki za svoje delovanje potrebujejo podatkovno bazo, datotečni sistem ali dostop do surovih podatkov na nivoju bloka. Na eno instanco lahko priklopimo več enot EBS.

Zelo uporabna funkcionalnost storitve EBS so posnetki (ang. snapshot). Posnetki so trenutna stanja virtualnih diskovnih enot. Ko prvič zajamemo posnetek, se v sistem za trajno shranjevanje podatkov (npr. Amazon S3) prenese celotna vsebina virtualnega diska. Ob naslednjem zajemu pa se prenesejo le razlike, glede na prejšnje stanje. Tako je vsak posnetek sestavljen iz osnovne kopije virtualnega diska (alfe) in enega ali več odsekov podatkov (delt), ki se razlikujejo od vsebine osnovne kopije. Tak način varnostnega kopiranja je bolj učinkovit in izboljša performanse, saj prenašamo manjše količine podatkov. Prav tako varčuje s prostorom in posledično zmanjša strošek najema pomnilniških kapacitet. Za varnostne kopije virtualnega diska lahko torej uporabimo posnetke, ki se shranjujejo v sistem S3. Iz posnetkov lahko hitro in enostavno ustvarimo novo enoto EBS in jo priklopimo na novo instanco. Prav tako lahko instanci odklopimo enoto EBS in jo priklopimo novi instanci. Delovanje je prikazano na sliki [4.3].

4.4 Varnost v EC2

Za varnost je pri EC2 poskrbljeno na več nivojih:

- Na nivoju operacijskega sistema gostitelja (ang. host OS)
- Na virtualni instanci ali gostujočem operacijskem sistemu (ang. guest OS)
- Na požarnem zidu
- Varnih podpisanih klicih API

Vsak od teh elementov gradi na zmogljivostih drugih. Ključni cilj je preprečiti, da bi podatke znotraj oblaka EC2 prestregel neavtoriziran uporabnik ali sistem in omogočiti instancam EC2 čim večjo varnost, brez da bi žrtvovali prilagodljivost in uporabniško svobodo pri nastavljanju sistema. Varnostni vidiki so povzeti po članku [2].

4.4.1 OS gostitelja (ang. host OS)

Administratorji znotraj Amazona za dostop do gostitelja virtualnih instanc uporabljajo večnivojsko avtentikacijo preko posebej za to vzpostavljenih orodij za vzdrževanje. Ta orodja so zasnovana z namenom, da zavarujejo glavno upravljalno jedro oblaka. Vsak tak administratorski dostop je zabeležen in pregledan. Ko Amazonov administrator ne potrebuje več dostopa do jedra gostiteljevega operacijskega sistema za točno določeno nalogo, se mu pravice za dostop do obravnavanih sistemov odvzamejo.

4.4.2 Gostujoči OS (ang. guest OS)

Virtualne instance so popolnoma pod nadzorom končnega uporabnika. Uporabniki imajo polni administratorski (ang. root) dostop do gostujočega operacijskega sistema in s tem nadzor nad uporabniškimi računi, storitvami in aplikacijami. Amazonovi administratorji nimajo dostopa do uporabniških instanc in se zato ne morejo prijaviti v gostujoči operacijski sistem. Amazon predlaga uporabo dobrih varnostnih praks, kot so izključitev možnosti dostopa samo na podlagi uporabniškega imena in gesla. Za dostop do instanc priporočajo neko obliko več nivojske avtentikacije, ali vsaj uporabo certifikatov in protokol SSH verzije 2 (SSHv2). Prav tako je priporočeno, da si uporabniki nastavijo sistem z različnimi nivoji pravic za različne skupine uporabnikov instanc. V primeru,

da je gostujoči operacijski sistem Linux, je priporočljivo preprečiti oddaljen administratorski dostop in do instance dostopati le s kombinacijo certifikata in neadministratorskega (ang. non-root) računa. Če nato potrebujemo administratorski dostop do gostujočega OS, lahko pravice pridobimo z ukazom *sudo*. Uporabniki morajo ustvariti svoje pare ključev za dostop do instance in s tem zagotoviti, da je par edinstven in ni deljen z drugimi strankami oblaka EC2.

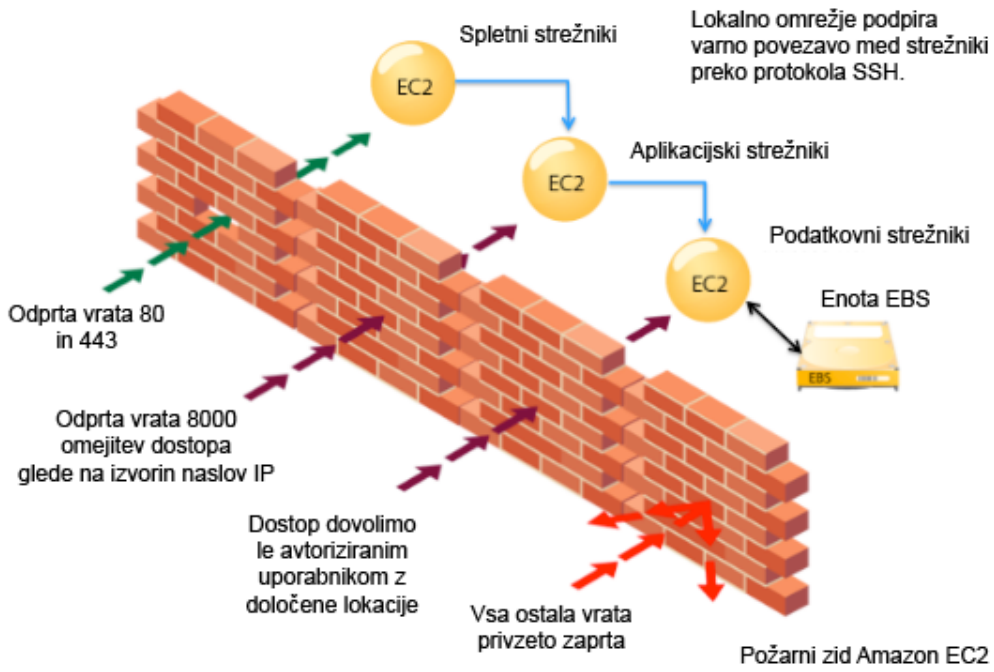
4.4.3 Požarni zid

Amazon EC2 omogoča uporabo celovitega požarnega zidu. Obvezen požarni zid je privzeto nastavljen na način "prepreči vse" (ang. deny-all) , ki preprečuje ves vhodni promet do instance. Uporabniki morajo eksplicitno odpreti vrata, ki jih uporabljajo za dostop do instance. Promet lahko omejimo glede na protokol in vrata, prav tako tudi glede na izvorni naslov IP, od koder prihaja promet. Določimo lahko posamezen IP ali razred naslovov IP iz katerih lahko dostopamo do instance.

Pravila požarnega zidu lahko shranimo v skupine, ki jih nato dodeljujemo različnim instancam. Za primer predstavimo klasično tri-nivojsko spletno aplikacijo v porazdeljenem sistemu strežnikov v javnem oblaku. Skupina spletnih strežnikov bi imela odprta vrata 80 (HTTP) in vrata 443 (HTTPS), če potrebujemo varno povezavo. Skupina aplikacijskih strežnikov bi imela odprta vrata 8000 (določeno v aplikaciji) in omejen dostop na podlagi izvornih naslovov IP. Skupini podatkovnih strežnikov pa bi odprli vrata 3306 (MySQL) in dostop dovolili le skupini aplikacijskih strežnikov. Vse tri skupine bi imele odprta vrata 22 (SSH), a le iz uporabnikovih naslovov IP. S takim mehanizmom zagotovimo visoko stopnjo varnosti spletne aplikacije. Delovanje požarnega zidu je prikazano na sliki [4.4].

Požarni zid deluje ločeno od gostujočega OS, za nastavljanje potrebujemo uporabniški certifikat X.509 in geslo za dostop do storitev AWS (Amazon Web Services), kar še zviša stopnjo varnosti.

Požarni zid AWS omogoča podrobne nastavitve in s tem uporabniku ponudi enostaven način za zagotovitev varnosti svojih instanc. Privzeta nastavitev požarnega zidu preprečuje ves vhodni promet, uporabnik pa mora sam premisliti, katera vrata in za kakšne namene bo odprl pot do svojih instanc. Vseeno pa se priporoča še dodane nastavitve varnosti na nivoju posameznih instanc. Primerne so uporabe požarnih zidov na gostujočih operacijskih sistemih kot npr. IPtables ali Windows Firewall ali VPN. S tem lahko omejimo vhodni in izhodni promet na vsaki instanci.



Slika 4.4: Požarni zid EC2

4.4.4 Klici vmesnika API

Klici vmesnika API EC2 za zaganjanje in ugašanje instance, za spreminjanje nastavitev požarnega zidu in klici podobnih funkcij morajo biti podpisani z uporabnikovim zasebnim ključem, ki ga prejmemo ob odprtju uporabniškega računa za dostop do Amazonovih storitev. Brez tega zasebnega ključa ni mogoče izvajati klicev programskega vmesnika v imenu lastnika ključa. Dodatno so klici API zavarovani s kriptirano povezavo SSL, ki zagotavlja zasebnost. Priporočeno je, da za klice API vedno uporabljamo s protokolom SSL zaščitene končne točke. Storitev AWS IAM[1] dodatno omogoča nastavljanje uporabniških pravic za klice API. Uporabniku lahko torej onemogočimo klice za nastavljanje požarnega zidu ali za zaganjanje in ugašanje instanc, dovolimo pa vse ostalo.

[1]AWS IAM (AWS Identity and Access Management) - dodatna storitev, ki omogoča ustvarjanje uporabniških računov znotraj osnovnega računa AWS

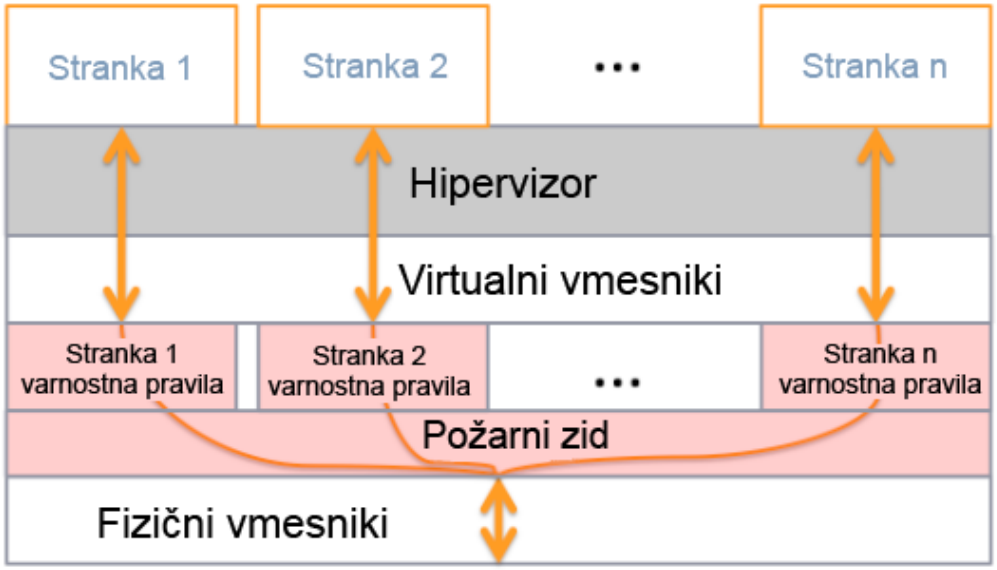
4.5 Virtualizacijsko okolje - hipervizor Xen

Amazon EC2 trenutno temelji na močno prilagojeni verziji hipervizorja Xen [16], ki izkorišča prednosti paravirtualizacije (v primeru gostovanja Linux). Paravirtualizacija virtualnim računalnikom preko vmesnika omogoča dostop do sistemskih virov gostitelja. Hipervizor skrbi za optimalno razporejanje virov med virtualnimi računalniki in tako zagotovi boljšo izkoriščenost strojne opreme. Gostujoči virtualni računalniki nimajo polnega dostopa do CPE in se v primerih, ko to potrebujejo, zanašajo na hipervizorja. CPE zagotavlja 4 različne načine pravic (ang. privilege modes): Ring 0-3. Način Ring 0 je najvišja stopnja priviligiranosti, Ring 3 pa najmanjša. Gostiteljev OS se izvaja v najbolj priviligiranem načinu Ring 0. Gostujoči OS se izvaja v manj priviligiranem načinu Ring 1, končne aplikacije pa v najmanj priviligiranem Ring 3 načinu. Taka ekplicitna virtualizacija fizičnih virov zagotavlja izolacijo gosta in gostitelja – hipervizorja, kar še dodatno pripomore k varnosti med obema.

Ločitev instanc

Različne instance, ki se izvajajo na istem fizičnem strežniku, so med seboj izolirane s pomočjo hipervizorja Xen. Amazon je prisoten v odprtokodni skupnosti, kar omogoča nenehno spremljanje in sodelovanje pri razvoju. Požarni zid AWS deluje tudi znotraj plasti hipervizorja, med fizičnim omrežnim vmesnikom in virtualnim vmesnikom, ki ga uporablja instanca. Vsak paket mora prečkati plast požarnega zidu. S tem zagotovimo, da imajo sosednje instance, ki gostujejo na istem fizičnem strežniku, popolnoma enak dostop med seboj, kot ga ima katerakoli druga omrežna naprava v medmrežju. Virtualni omrežni vmesnik lahko obravnavamo kot fizičnega priključenega v medmrežje. Na podoben način je izoliran tudi fizični pomnilnik.

Uporabnikove instance nimajo neposrednega dostopa do fizičnih diskovnih naprav, dodeljeni so jim virtualni diski. Plast diskovne virtualizacije za vsak blok uporabniških podatkov samodejno poskrbi, da ni nikoli viden ostalim uporabnikom EC2. Seveda obstaja možnost, da podatke delimo z ostalimi, če to želimo. Amazon za varnostno kritične podatke priporoča še dodatne varnostne ukrepe. Primerna rešitev za dodatno zaščito podatkov je uporaba kriptiranega datotečnega sistema na virtualni diskovni enoti.



Slika 4.5: Ločitev instanc

Poglavje 5

Integracija hibridnega oblaka

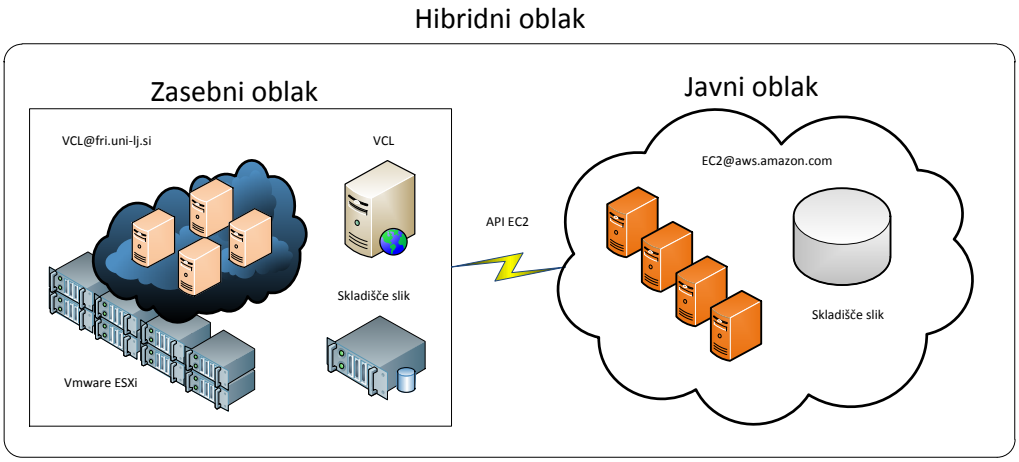
V tem poglavju bomo predstavili tehnično osnovo, ki je potrebna za razumevanje delovanja modula za oskrbovanje z viri iz javnega oblaka. Na primeru Laboratorija za računalniške komunikacije bomo opisali arhitekturo hibridnega oblaka, ki vsebuje zasebni in javni oblak. Trenutno stanje sestoji le iz zasebnega oblaka, predlagali bomo rešitev hibridnega oblaka s povezavo v javni oblak Amazon EC2. Za povezavo med zasebnim in javnim oblakom bomo uporabili vmesnik API javnega oblaka Amazon EC2. Na kratko bomo predstavili način uporabe vmesnika in predstavili knjižnico, ki smo jo uporabili za namen povezovanja z EC2.

Za razvoj novega modula za oskrbovanje z viri javnega oblaka smo najprej namestili testno okolje VCL, pojasnili bomo podrobnosti namestitve in arhitekturo razvojnega okolja, ki sestoji iz treh enot: spletnega vmesnika, podatkovne baze in upravljalca VCL. Seznanili se bomo z modularno arhitekturo in objektnim pristopom upravljalnega dela VCL, kar je osnova za razumevanje načina gradnje novih modulov VCL. Z razrednim diagramom bomo predstavili strukturo modulov in v diagram umestili naše nove module, ki so potrebni za integracijo hibridnega oblaka. Ogledali si bomo podrobno zgradbo modula in opisali njegove metode ter potek delovanja procesa nalaganja in zajemanja slike. Prikazali bomo primer uporabe z vidika končnega uporabnika. Na koncu poglavja kritično ocenimo našo rešitev in predstavimo prednosti in slabosti hibridnega oblaka.

5.1 Arhitektura hibridnega oblaka

Za integracijo hibridnega oblaka smo morali najprej oceniti trenutno stanje zasebnega oblak, ki poganja okolje VCL. V Laboratoriju za računalniške komunikacije imamo 11 strežnikov z VMware ESXi 4 virtualizacijsko tehnologijo, ki omogoča zaganjanje virtualnih računalnikov. Na dodaten 12. strežnik je nameščeno okolje VCL s spletnim vmesnikom, podatkovno bazo in upravljalcem VCL. Vsak strežnik ima 4-jedrni procesor in 8 GB delovnega pomnilnika. Trenutno je okolje nastavljeno tako, da na enem strežniku poganjamo približno 30 virtualnih računalnikov s 256 MB delovnega pomnilnika. Ker imajo strežniki omejeno količino delovnega pomnilnika, bi se ob povečanju zmogljivosti virtualnih računalnikov število možnih instanc zmanjšalo (npr. 15 virtualnih računalnikov s 512 MB delovnega pomnilnika). To omejitev bomo navidezno odpravili s povezavo z javnim oblakom, kjer so kapacitete neomejene. Slike virtualnih računalnikov so shranjene v skladišču slik, ki se na nahaja na omrežno priklopljenem pomnilniku (ang. Network Attached Storage - NAS). Pri vajah uporabljamo množico slik, ki zajemajo operacijske sisteme Windows Server 2003, FreeBSD in Debian. Študenti prejmejo polni administratorski dostop do virtualnih okolij, po koncu rezervacije pa se okolja zbrišejo in morajo študentje sami poskrbeti za shranjevanje lastnih podatkov.

Predlagana rešitev arhitekture hibridnega oblaka je prikazana na sliki [5.1]. Za povezovanje z javnim oblakom na obstoječe vozlišče z upravljalcem VCL namestimo nov modul za oskrbovanje z viri javnega oblaka in s tem omogočimo možnost zaganjanja virov v javnem oblaku. Kot ponudnika javnega oblaka smo izbrali Amazon EC2, ki je trenutno med vodilnimi na področju računalništva v oblaku in ponuja storitveni model IaaS, ki ustreza potrebam Laboratorija za računalniške komunikacije, saj želimo, da se študentje spoznajo z nižjenivojskim delovanjem računalnikov, kar model SaaS ne omogoča. Zasebni in javni oblak sta povezana preko vmesnika API EC2, ki omogoča nadziranje virtualnih računalnikov (zaganjanje, ugašanje, preverjanje) in ustvarjanje slik. Obstoječe slike lahko razmeroma enostavno preselimo v javni oblak, ki ima za to predvideno svoje skladišče slik. Slednje je smiselno, saj je prenašanje slik iz zasebnega v javni oblak dolgotrajno, ker gre za večje količine podatkov, ki se prenašajo preko medmrežja. Ko pa se slike nahajajo v javnem oblaku, je manipulacija z njimi hitra in enostavna. Z javnim oblakom odpravimo omejitve zasebnega, saj lahko uporabljamo navidezno neomejeno količino virov in plačujemo najemnino za dejansko porabo.



Slika 5.1: Arhitektura hibridnega oblaka v Laboratoriju za računalniške komunikacije

5.2 Povezava z javnim oblakom

5.2.1 Vmesnik API Amazon EC2

Vmesnik API Amazon EC2 je spletna storitev, ki omogoča zaganjanje in upravljanje virtualnih računalnikov z operacijskimi sistemi Linux oziroma Windows v Amazonovih podatkovnih centrih. V skladu s storitveno usmerjeno arhitekturo uporabnikom zagotavlja popoln nadzor nad računalniškimi viri v javnem oblaku. Virtualne računalnike lahko preko spletnih storitev zaganjamo, ustavljamo, ustvarjamo slike, shranjujemo trenutna stanja itd. Amazon pa nam izda račun na podlagi dejanske uporabe njihovih virov.

Amazon EC2 zagotavlja dva vmesnika API na podlagi dveh protokolov: SOAP in REST. Vmesnika se razlikujeta le po klicih, saj oba vračata enake dokumente XML. Amazon vmesnika stalno nadgrajuje in dodaja nove funkcionalnosti. Opis vseh funkcionalnosti lahko najdemo v dokumentu WSDL, ki je na voljo na spletu (EC2 WSDL). Podroben opis vseh funkcionalnosti vmesnika API najdemo v dokumentaciji [11], spodaj pa povzamemo le ključne.

PRIMER UPORABE vmesnika API na podlagi protokola REST
Poizvedbeni zahtevki so zahteve HTTP oziroma HTTPS, ki jih pošljemo z GET ali POST načinom s parametrom “action”, ki definira metodo. Ne glede na to, kateri protokol uporabimo, mora biti vsak klic API podpisan z uporab-

nikovim zasebnim ključem. Podpis se izračuna na podlagi vsebine zahtevka in zasebnega ključa. V klic API ga dodamo z atributom 'Signature'. Varnost vmesnika API smo obravnavali tudi v poglavju 4.4.4.

Struktura zahtevka GET

Končna točka – dostopna točka spletne storitve (npr. ec2.amazonaws.com, ki je privzeto nastavljeno na regijo us-east-1)

Akcija – metoda, ki jo želimo izvesti (npr. zaženi virtualni računalnik (*RunInstance*) ali opiši sliko (*DescribeImage*))

Atributi – atributi, ki jih podamo z metodo (npr. identifikator virtualnega računalnika (*InstanceID*) ali število virtualnih računalnikov, ki jih želimo zagnati (*MaxCount*))

Primer zahtevka GET za opis aktivnih instanc je prikazan v izvorni kodi [5.1]. Primer odziva XML na omenjeni zahtevek pa v [5.2].

Izvorna koda 5.1 Primer zahtevka GET za opis aktivnih instanc

```
https://ec2.amazonaws.com/
?AWSAccessKeyId=AKIAICSEUY3YGJHGNK4Q
&Signature=boKPO0gHNFL0p3uSy%2B3IhMjXV6jP3SVnuj9%2FK9HR
&Version=2010-06-15
&Timestamp=2011-08-22T17%3A22%3A38.000Z
&Action=DescribeKeyPairs
&SignatureMethod=HmacSHA256
&SignatureVersion=2
```

5.2.2 Perlova knjižnica VM::EC2

Komunikacijo z oblakom EC2 si lahko olajšamo s knjižnicami, ki implementirajo vmesnik Amazon EC2 API. Namesto, da sami pišemo zahtevke REST ali SOAP, uporabimo knjižnice, ki omogočajo enostavnejši dostop do vmesnika in pospešijo razvoj. Amazon zagotavlja knjižnice za vse najbolj pogoste programske jezike, prav tako pa knjižnice najdemo v odprtokodni skupnosti. Pri izdelavi modula za oskrbovanje z viri EC2 smo se omejli na knjižnice v programskem jeziku Perl, saj je celoten upravljalni del okolja VCL spisan v

Izvorna koda 5.2 Primer odziva XML na zgornji zahtevek GET

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeKeyPairsResponse
xmlns="http://ec2.amazonaws.com/doc/2011-07-15/">
 <requestId>7a62c49f-347e-4fc4-9331-6e8eEXAMPLE
 </requestId>
  <keySet>
    <item>
      <keyName>gsg-keypair</keyName>
      <keyFingerprint>
          1f:51:ae:28:bf:89:e9:d8:1f:25:
          5d:37:2d:7d:b8:ca:9f:f5:f1:6f
      </keyFingerprint>
    </item>
  </keySet>
</DescribeKeyPairsResponse>
```

omenjenem jeziku. Najbolj primerno smo izbrali v skladišču Perlovih knjižnic CPAN. Knjižnica VM::EC2 [13] implementira vse nujne funkcionalnosti in se stalno razvija. Zato podpira tudi najnovejše funkcionalnosti vmesnika API EC2, kot je npr. dodajanje poljubnih oznak virom v EC2, s čimer lahko poenotimo poimenovanja posameznih virov čez celoten hibridni oblak. Oznake virtualnih računalnikov, slik, uporabnikov so tako lahko enotne tako v zasebnem kot tudi v javnem oblaku. To olajša vzdrževanje sistema in iskanje morebitnih napak. Knjižnica je zasnova tako, da bo v prihodnjih verzijah omogočala povezavo z zasebnim oblakom Eucalyptus[1], ki uporablja podoben vmesnik API za dostop do njegovih virov.

Knjižnica je objektno usmerjena in dobro dokumentirana, zato je delo z njo enostavno in hitro. V izvorni kodi [5.3] je prikazan primer uporabe knjižnice VM::EC2.

Razvidno je, kako se povežemo z javnim oblakom s pomočjo javnega (*access_key*) in zasebnega ključa (*secret_key*). Knjižnica pri vsakem klicu vmesnika API EC2 poskrbi za izračun podpisa, ki temelji na vsebini zahtevka in zasebnem ključu. Nato pridobimo nekaj podrobnosti o sliki z identifikatorjem "ami-12345" in zaženemo dva virtualna računalnika z omenjeno sliko. Počakamo,

[1]Eucalyptus - odprtokodna rešitev za implementacijo zasebnega oblaka (`http://www.eucalyptus.com/`)

Izvorna koda 5.3 Primer uporabe knjižnice VM::EC2

```
#ustvari nov VM::EC2 objekt
my $ec2 = VM::EC2->new(-access_key=>'AKIAICSEUY3Y',
-secret_key=>'KQMTEVFBR08OoawRctvWDo',
-endpoint=>'http://ec2.amazonaws.com');

#pridobi podatke o sliki z identifikatorjem
my $image = $ec2->describe_images('ami-12345');

#izpisi podatke o sliki
my $architecture = $image->architecture;
my $description  = $image->description;
my @devices      = $image->blockDeviceMapping;
for my $d (@devices) {
 print $d->deviceName,"\n";
 print $d->volumeSize,"\n";
}

#zazeni dve instanci
my @instances = $image->run_instances(-key_name=>'My_key',
-security_group=>'default',-min_count=>2,
-instance_type=>'t1.micro')
or die $ec2->error_str;

#pocakaj, da se instanci zazeneta
$ec2->wait_for_instances(@instances);

#izpisi trenutno stanje in ime DNS obeh instanc
for my $i (@instances) {
 my $status = $i->current_status;
 my $dns    = $i->dnsName;
 print "$i: [$status] $dns\n";
}

#oznaci obe instanci z Vloga="streznik"
foreach (@instances) {$_->add_tag(Vloga=>'streznik');

#ustavi obe instanci
foreach (@instances) {$_->stop}
```

da se zaženeta in preverimo njun status in javni naslov IP. Računalnikoma dodamo še oznaki in ju na koncu ustavimo. Primer predstavlja tipičen proces življenja virtualnih računalnikov v javnem oblaku.

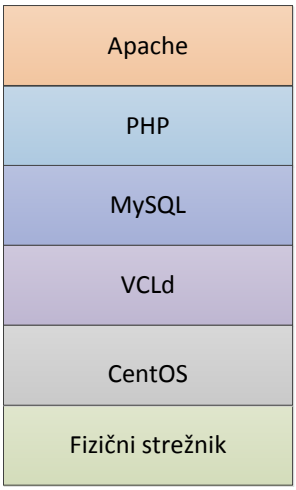
5.3 Namestitev okolja VCL za potrebe razvoja

Za potrebe razvoja modula za oskrbovanje z viri EC2 smo najprej namestili okolje VCL. Vzeli smo najnovejšo Apache VCL 2.2.1 verzijo. Na fizični strežnik smo namestili operacijski sistem CentOS 5.6. Linux distribucija CentOS je popolnoma odprt operacijski sistem in je sorodna z Red Hat Enterprise distribucijami, ki so plačljive. Razvijalci sistema VCL priporočajo uporabo operacijskega sistema na osnovi Red Hat verzij. Tudi sami večinoma uporabljajo CentOS, vseeno pa je VCL možno namestiti na katerokoli drugo verzijo operacijskega sistema Linux. Samo jedro okolja VCL je sestavljeno iz treh delov: (a) spletnega vmesnika, (b) podatkovne baze in (c) upravljalca VCL. Vsak del lahko izvajamo na samostojenemu strežniku, za testno okolje pa smo vse naložili na isti strežnik. Za spletni vmesnik smo nastavili strežnik Apache in okolje PHP z vsemi potrebnimi knjižnicami. Prav tako smo namestili MySQL strežnik in ustvarili podatkovno bazo, ki vsebuje vse potrebno za delovanje okolja VCL. Največ pozornosti smo namenili upravljalcu VCL, ki je zaledni sistem in skrbi za oskrbovanje z viri. Glavni del upravljalca VCL je demonska storitev VCLd, ki se zažene ob zagonu sistema in neprestano nadzira celoten sistem. Njene naloge so npr. preverjanje rezervacij, zaganjanje virtualnih računalnikov, ustavljanje virtualnih računalnikov po preteku rezervacij, izvajanje procesa zajema slik itd. Na sliki [5.2] je razviden sklad razvojnega okolja VCL. Nad fizičnim strežnikom je plast operacijskega sistema CentOS, nato VCLd demonska storitev, za podatkovno bazo skrbi plast MySQL, za spletni vmesnik pa strežnik Apache in izvajalno okolje PHP.

5.4 Modularna arhitektura VCL

5.4.1 Ozadje

Zaledni sistem VCL je bil z verzijo 2.0 temeljito predelan in temelji na modularnem ogrodju. Modularnost omogoča ločitev posameznih delov kode od samega jedra. Na podlagi obstoječih modulov lahko gradimo nove. S tehniko dedovanja prevzamemo metode podedovanega modula, jih dopolnimo ali spremenimo ter dodamo nove.



Slika 5.2: Sklad razvojnega okolja VCL

VCL vzajemno deluje z zunanjimi tehnologijami kot so:

- Sistemi za oskrbovanje viri (ang. provisioning engines)
- Operacijskimi sistemi
- Orodji za nadzor računalniških virov

Katere tehnologije se uporablja, je odvisno od posamezne namestitve okolja VCL. Ravno moduli omogočajo prilagajanje različnim tehnologijam brez posega v samo jedro sistema. Prednosti modulov niso omejene samo na zunanje tehnologije. Moduli se uporabljajo za implementacijo notranjih metod VCL, ki pa se lahko razlikujejo glede na nastavitve okolja. Lahko razvijemo dva popolnoma različna algoritma, ki previdevata, katero sliko bi bilo nabolj smiselno naložiti na sproščeni računalnik po preteku rezervacije. Cilj algoritma je čim bolj natančno ugotoviti, katero sliko bo na sproščenem računalniku želel zagnati naslednji uporabnik. Algoritma zapakiramo v dva ločena modula, nato preko spletnega vmesnika nastavimo, katerega bomo uporabljali na posameznem upravljalnem vozlišču. S tem zagotovimo prožnost sistema in olajšamo vzdrževanje.

5.4.2 Objektna usmerjenost in dedovanje

Modularno ogrodje VCL izkorišča prednosti objektno usmerjenega programiranja in dedovanja. To omogoča dedovanim modulom dostop do metod in

atributov podedovanih razredov. S tem razbremenimo razvijalca, saj se mu ni potrebno ukvarjati s podrobnostmi celotnega sistema (npr. pisanje stavkov SQL za dostopanje do podatkov iz baze) in mu omogočimo, da se osredotoči le na gradnjo svojega modula. S tem rešimo problem podvajanja kode in vzpodbudimo ponovno rabo že napisane kode. Prav tako močno olajšamo posodabljanje sistema VCL, saj ne posegamo v kodo, ki bi jo lahko razvijalci v prihodnjih verzijah spremenili.

Cilj diplomskega dela je VCL povezati z javnim oblakom. Za ta namen smo razvili modula *ec2.pm* in *LinuxEC2.pm*. Na sliki [5.3] je prikazan razredni diagram modulov VCL in vsebuje na novo razvita modula. Pri gradnji modulov smo bili pozorni na to, da smo čim večji del metod pustili čim višje v hierarhiji razrednega diagrama. Tako smo v novih modulih implementirali le metode, ki so bile nujno potrebne zaradi posebnosti javnega oblaka. Tako smo pri *LinuxEC2.pm* modulu povozili (ang. override) le metodi *sanitize()* in *update_public_ip()*, vse ostale deduje od razreda *Linux.pm*.

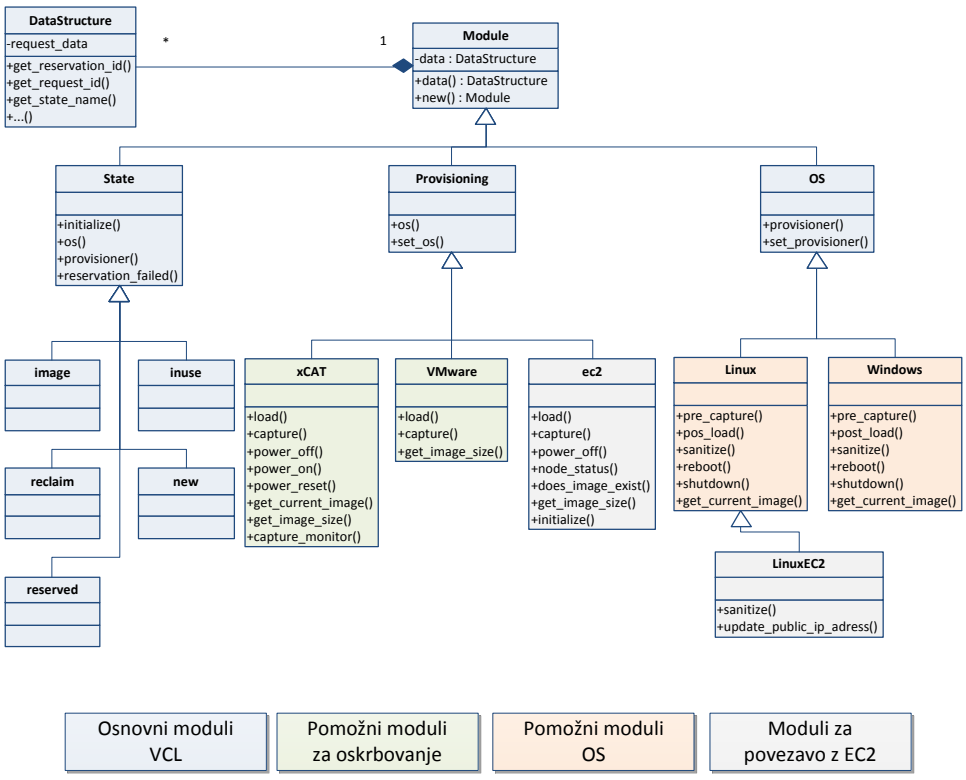
V razrednem diagramu modri moduli predstavljajo osnovne module VCL, ki sestavljajo samo jedro sistema. Rdeči in zeleni predstavljajo pomožne module, ki so na voljo v VCL. Siva modula predstavljata rezultat tega diplomskega dela in omogočata povezavo z javnim oblakom Amazon EC2.

Osnovni moduli pomožnim omogočajo naslednje funkcionalnosti:

- Privzete konstruktorje
- Dostop do podatkov
- Inicializacijo modulov
- Dostop do drugih pomožnih modulov, kjer je to primerno (npr. moduli *OS* lahko dostopajo do *Provisioning* modulov)

5.5 Razvoj modula za oskrbovanje

Z vzpostavljenim testnim okoljem VCL in razumevanjem modularne arhitekture sistema smo se lahko lotili razvoja modula za oskrbovanje z viri javnega oblaka Amazon EC2. Za razvojno orodje smo izbrali Eclipse IDE (`http://www.eclipse.org/`) z dodatkom EPIC (`http://www.epic-ide.org/`), ki omogoča razvoj v programskem jeziku Perl. Najprej smo po priporočilih dokumentacije VCL [8] definirali vmesnik modula za oskrbovanje, preučili postopke nalaganja in zajemanja slik ter se nato lotili programiranja.



Slika 5.3: Razredni diagram VCL

5.5.1 Vmesnik modula za oskrbovanje

Modul zajema naslednje metode:

initialize() Metoda se zažene vsakič, ko upravljalec VCL ustvari nov izvod objekta modula za oskrbovanje. Uporablja se za spremljanje poteka procesa oskrbovanja.

node_status() Metoda preveri trenutno stanje obravnavanega vozlišča. Preko nje se modul zaveda, v kakem stanju je vozlišče, ali je dosegljivo in ima naloženo pravo sliko. Metoda vrača zgoščeno tabelo z različnimi informacijami o trenutnem stanju vozlišča.

capture() Metoda zajame trenutno stanje vozlišča. To pomeni, da v obliki slike shrani celotno vsebino trdega diska v skladišče slik. Meta podatki o sliki se shranijo v podatkovno bazo in je tako na voljo za prihodnjo uporabo. Tipičen primer uporabe je ob nameščanju nove programske opreme na obstoječo sliko, ki jo nato shranimo kot novo in omogočimo prihodnjo uporabo. Metoda je odgovorna za klic metode *pre_capture()* modula *OS*, s katero pripravimo okolje na shranjevanje slike (npr. brisanje podatkov o trenutno prijavljenem uporabniku). Prav tako mora zagotoviti, da se celoten proces zajema slike v skladišče slik uspešno zaključi. Sam proces lahko traja dlje časa, saj gre za prenašanje večjih količin podatkov (vsebine celotnega trdega diska). V kolikor pride do napake, metoda vrne ustrezno napako.

- pričakovano začetno stanje:
  - Administrator je na obravnavano vozlišče uspešno namestil novo programsko opremo oziroma spremenil sliko na poljuben način in preko spletnega vmesnika zagnal proces zajema nove slike.
  - Vozlišče s sliko, ki jo želimo zajeti, je aktivno in dostopno.
  - Modul ima dostop do podatkovnega objekta z vsemi ustreznimi metapodatki o sliki, ki še ne obstaja in bo zajeta.
- pričakovano končno stanje:
  - Celoten proces zajema slike se je uspešno zaključil oziroma se je pojavila napaka.
  - Metoda je počakala, da se je slika shranila v skladišče slik in je na voljo.

- metodo pokliče:
  - *image.pm::process();*
- metoda vrne vrednost:

  1 Slika je uspešno zajeta, image.pm lahko nadaljuje proces zajema s posodabljanjem podatkov o novi sliki in zagotovi njeno dostopnost.

  0 Prišlo je do napake, administrator mora raziskati, kaj se je zgodilo.

load() Metoda poskrbi za zagon virtualnega računalnika z ustrezno sliko. Na podlagi vhodnih podatkov preveri obstoj slike in zažene ustrezno instanco v javnem oblaku. Prav tako poskrbi za posodobitev naslovov IP v podatkovni bazi, saj javni oblak virtualnemu računalniku ob vsakem zagonu dinamično dodeli nov javni naslov IP.

- pričakovano začetno stanje:
  - V podatkovni bazi obstaja računalnik in slika, ki ju želimo zagnati.
  - Modul ima dostop do javnega oblaka in pravice za zaganjanje virtualnih računalnikov.
- pričakovano končno stanje:
  - Celoten proces zaganjanja virtualnega računalnika z ustrezno sliko se je uspešno zaključil oziroma se je pojavila napaka.
  - Metoda je počakala, da se je virtualni računalnik v javnem oblaku zagnal, je aktiven in dostopen preko protokola SSH.
- metodo pokliče:
  - *new.pm::process();*
- metoda vrne vrednost:

  1 Virtualni računalnik se je uspešno zagnal, *new.pm* lahko nadaljuje proces in na virtualnem računalniku ustvari nov uporabniški račun in uporabniku omogoči dostop.

  0 Prišlo je do napake, administrator mora raziskati, kaj se je zgodilo

power_off() Metoda ustavi delovanje virtualnega računalnika. Ob tem se izgubijo vsi podatki, zato mora uporabnik sam poskrbeti za shranjevanje svojih podatkov.

does_image_exist() Metoda preveri obstoj slike v javnem oblaku.

get_image_size() Metoda vrne velikost v GB, ki jo zaseda posameza slika v oblaku.

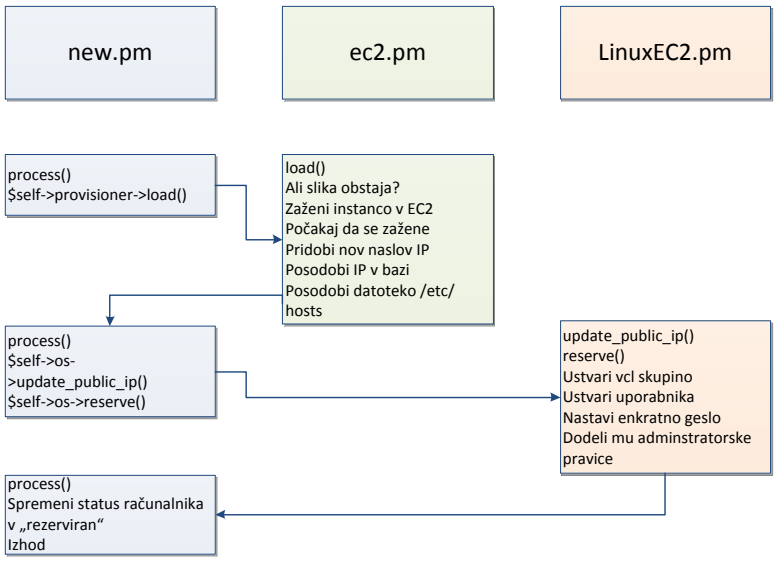
5.5.2 Potek nalaganja nove slike

Jedro upravljalca VCL je demonska storitev VCLd. Storitev se zažene ob zagonu upravljalnega vozlišča in neprestano preverja, kaj je potrebno storiti. Vsakih 10 sekund preveri, ali obstaja nova rezervacija, ki jo je potrebno sprocesirati. Ko jo najde, s pomočjo *utils.pm::get_request_info()* zbere vse informacije o rezervaciji in ustvari *DataStructure* objekt. Nato ustvari *State* objekt, ki opisuje trenutno stanje obravnavane rezervacije. Pri nalaganju nove slike se ustvari modul *new.pm* (ostali možni so *inuse.pm, image.pm, reclaim.pm*). Metoda *State.pm::initialize()* ustvari ustrezen objekt *OS* in objekt modula za oskrbovanje glede na podatke, ki so podani z rezervacijo (v našem primeru *LinuxEC2.pm* in *ec2.pm*). Ko so vsi potrebni objekti ustvarjeni, VCLd ustvari nov podproces za novo ustvarjeni *State* objekt. Ta podproces pokliče metodo *process()*, ki izvede vse potrebno za procesiranje rezervacije in vrne obvestilo o uspešnem ali neuspešnem zaključku.

Kot je razvidno v diagramu [5.4], modul *new.pm* sproži metodo *load()* v našem *ec2.pm* modulu. Proces preveri, ali slika obstaja in v primeru pozitivnega odgovora zažene nov instanco v oblaku EC2. Počaka, da se instanca zažene in pridobi njen naslov IP ter ustrezno posodobi podatke v bazi VCL in */etc/hosts* datoteki. Po uspešnem zaključku procesa *load()*, delo prevzame modul *LinuxEC2.pm*, ki ustvari novega uporabnika na virtualnem računalniku in mu dodeli enkratno geslo, ki velja le za trenutno rezervacjo. Prav tako mu dodeli administratorske pravice.

5.5.3 Potek zajema nove slike

Ko storitev VCLd sprejeme zahtevek za zajem nove slike, se ustvari nov *State* objekt, konkretno modul *image.pm*. Potek je prikazan na diagramu [5.5]. Modul najprej uporabniku, ki je zahteval zajem, pošlje sporočilo o začetku zajema nove slike. Na podlagi vhodnih podatkov pokliče metodo *capture()* objekta *ec2.pm*. Ta s pomočjo *LinuxEC2.pm* objekta sproži postopek *pre_capture()*,



Slika 5.4: Diagram poteka nalaganja slike

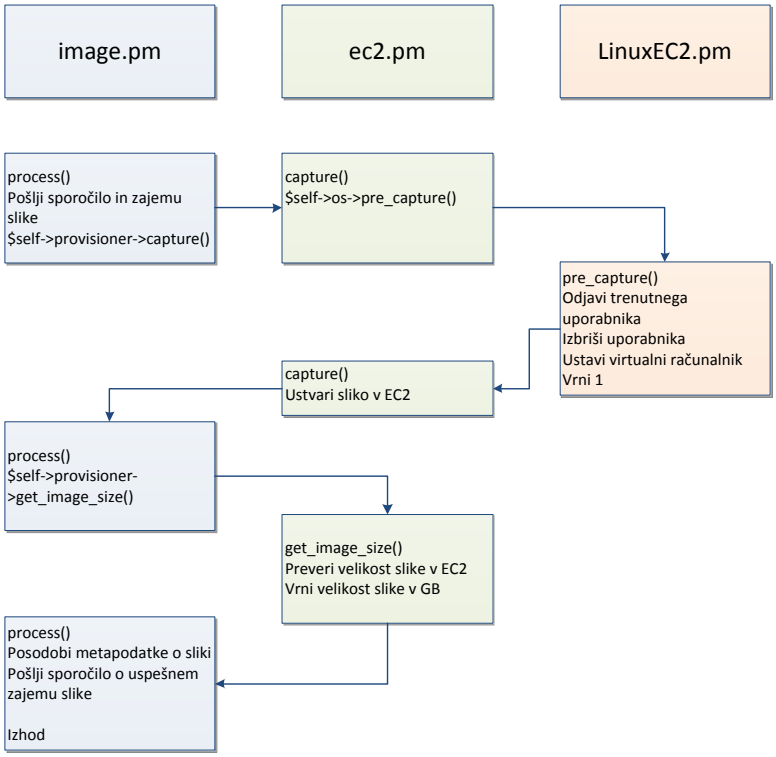
ki poskrbi, da je slika pripravljena za zajem. Ko ta konča s predpripravo, *ec2.pm* ustvari sliko v javnem oblaku. Objektu *image.pm* vrne potrditev o uspešnem zajemu slike, ta pa nato še preveri velikost novo nastale slike in posodobi podatke v podatkovni bazi VCL.

5.6 Primer uporabe končnega uporabnika

Funkcionalnosti okolja VCL ostajajo enake, ne glede na to, ali uporabljamo vire iz zasebnega ali javnega oblaka. Z vidika končnega uporabnika bomo prikazali primera uporabe virov iz javnega oblaka. Prikazali bomo postopek rezervacije virtualnega računalnika in zajema nove slike na podlagi obstoječe osnovne slike.

5.6.1 Rezervacija virtualnega računalnika

Uporabnik do okolja VCL dostopa preko spletnega vmesnika. Prijavi se z dodeljenim uporabniškim imenom in geslom. Povezava do spletnega vmesnika je kriptirana (protokol SSL). Po uspešni prijavi se uporabniku izriše uporabniški vmesnik glede na njegove pravice. Za primer rezervacije smo se prijavili z navadnim uporabnikom, ki ima pravico do uporabe virov VCL, ne more pa posegati



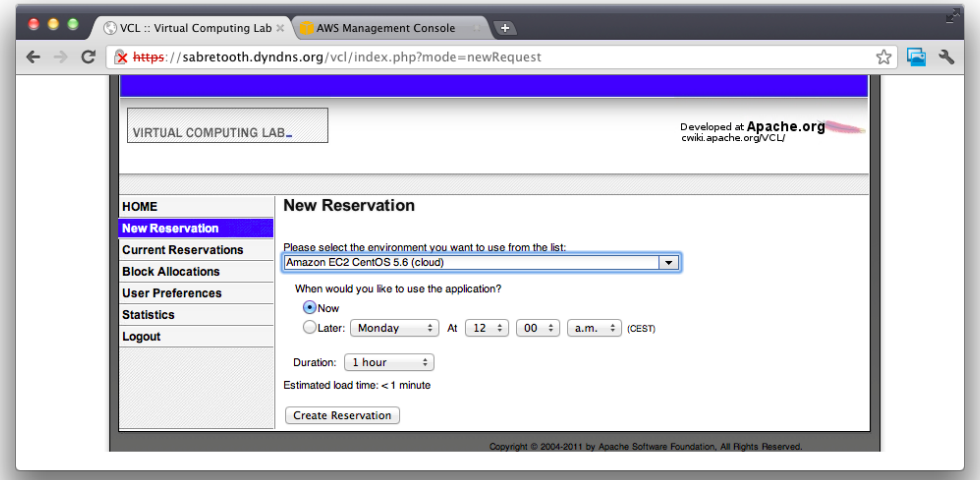
Slika 5.5: Diagram poteka zajema nove slike

v administracijo sistema. Kot je prikazano na sliki [5.6], se na podstrani "New Reservation" prikažejo slike, do katerih lahko dostopa.

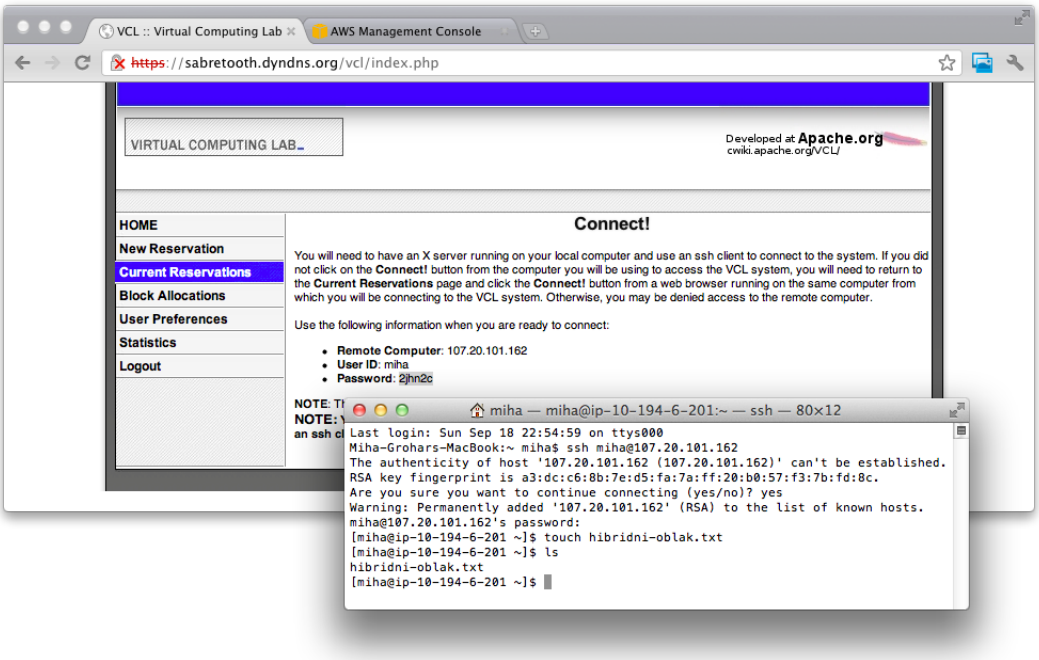
Uporabnik izbere željeno sliko v javnem oblaku in počaka, da sistem zažene virtualni računalnik. V primeru oblaka EC2 to traja približno eno minuto. Nato spletni vmesnik sporoči, da je rezervacija potrjena in uporabniku preda dostopne podatke. Za dostop do virtualnega računalnika potrebujemo njegov javni naslov IP, uporabniško ime in geslo. Na sliki [5.7] je prikazana uspešno zaključena rezervacija in povezava na virtualni računalnik preko protokola SSH.

5.6.2 Zajem nove slike

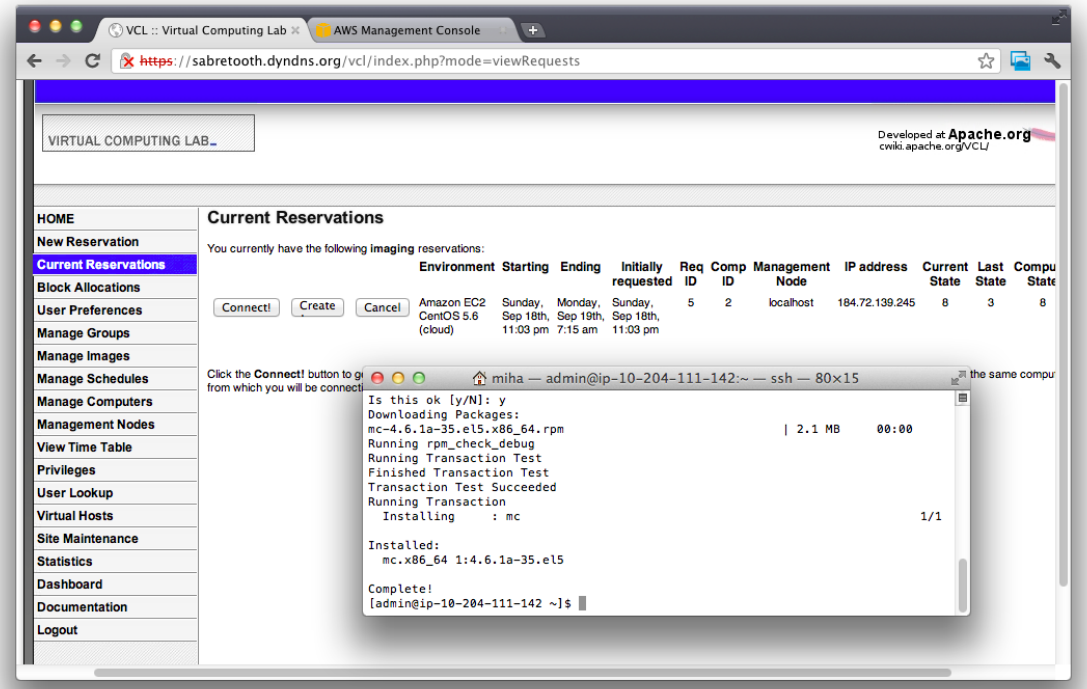
Uporabnik z administratorskimi pravicami lahko na podlagi obstoječe slike ustvari novo. Zato je potrebno zagnati virtualni računalnik z obstoječo sliko, nanj naložiti novo programsko opremo oziroma prilagoditi okolje po svojih željah in nato sprožiti proces zajema nove slike. Uporabnik z ustreznimi pravi-



Slika 5.6: Izbira slike



Slika 5.7: Uspešna rezevacija in povezava preko protokola SSH

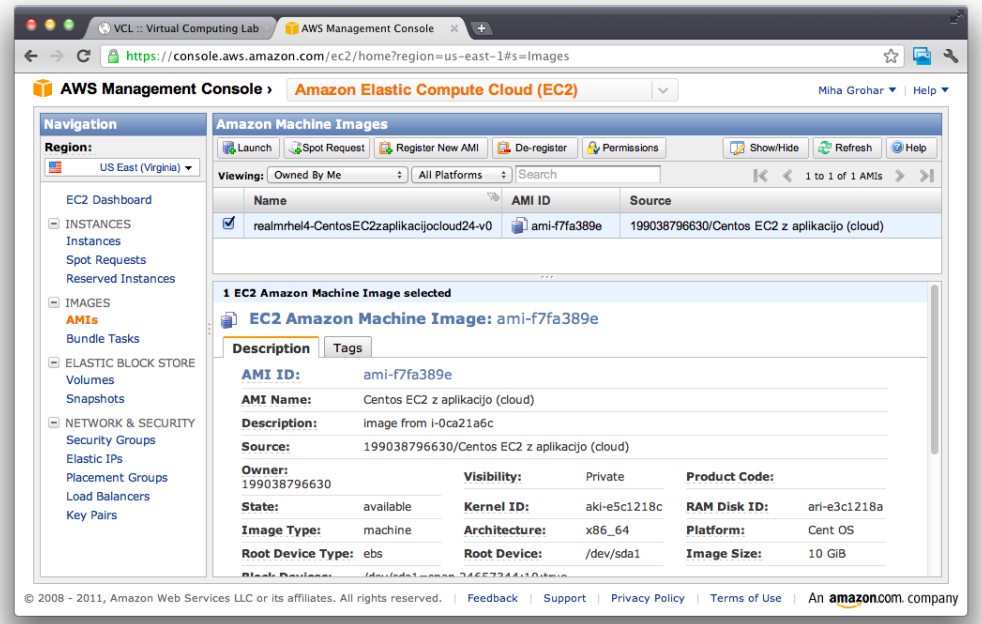


Slika 5.8: Zajem nove slike

cami postopek začne s posebno rezervacijo virtualnega računalnika namenjeno zajemu nove slike. Ob ustvarjanju rezervacije za zajem slike se nam prikaže tudi obrazec, v katerega vnesemo podatke o novi sliki. Podatki zajemajo ime slike, opis in po želji še ostale. Postopek rezervacije je enak navadnemu, ko pa se uporabnik enkrat prijavi na rezervirani računalnik, se mu v spletnem vmesniku ponudi možnost "Create Image", kar je prikazano na sliki [5.8]. S pritiskom na gumb sprožimo proces zajema nove slike v javnem oblaku. Postopek zajema lahko traja nekaj minut, odvisno od velikosti slike. Ko se proces zaključi, uporabnik prejme obvestilo o uspešnem zajemu nove slike, ki je tako pripravljena za uporabo. Na sliki [5.9] je prikazan spletni vmesnik javnega oblaka EC2, kjer si lahko pogledamo podrobnosti pravkar zajete slike.

5.7 Prednosti in slabosti hibridnega oblaka

Po uspešni integraciji hibridnega oblaka, smo kritično ocenili njegovo delovanje ter identificirali njegove prednosti in slabosti.



Slika 5.9: Prikaz zajete slike v spletnem vmesniku Amazon EC2

Prednosti:

- hitra vzpostavitev zelo zmogljive infrastrukture
- ni začetnih stroškov
- hitro prilagajanje potrebam
- boljša izkoriščenost virov
- manj administrativnega in vzdrževalnega dela
- manjša poraba energije v zasebnem oblaku
- možnosti visoko zmogljivega računalništva (HPC)

S povezavo z javnim oblakom pridobimo navidezno neomejene zmogljivosti. Zelo hitro lahko vzpostavimo zmogljivo virtualno infrastrukturo, za katero bi drugače potrebovali veliko denarja, časa in prostora. Vstopnih stroškov pri javnem oblaku ni, plačujemo le dejansko porabo. Povečanim potrebam po

virih se lahko hitro prilagodimo in povečamo zmogljivosti hibridnega oblaka, prav tako jih enostavno zmanjšamo, ko se zahteve po virih umirijo. S tem v dosežemo boljšo izkoriščenost virov, kot bi jo v primeru nadgradnje obstoječega zasebnega oblaka. Pomembna prednost javnega oblaka je ta, da povsem odpade vzdrževanje fizične strojne opreme. V primerih odpovedi fizične strojne opreme v javnem oblaku (npr. okvara trdega diska), se končni uporabnik tega ne zaveda. Posledično se z uporabo virov javnega oblaka zmanjša tudi poraba energije v zasebnem oblaku, saj je ta manj obremenjen.

Slabosti:

- nevarnost podražitev storitev javnega oblaka
- vkleščenje podatkov (ang. data lock-in), prevelika odvisnost od enega samega ponudnika
- slaba interoperabilnost med javnimi oblaki, saj standardni vmesniki še niso določeni
- izpadi delovanja in nedostopnost storitev
- ponudnik javnega oblaka lahko propade

Ker nobena rešitev ni brez kompromisov, smo preučili tudi slabosti. Z zavedanjem, kakšna tveganja obstajajo v javnem oblaku, lahko bolj preudarno uporabljamo rešitev hibridnega oblaka. Kot pri vseh storitvenih poslovnih modelih, tudi v javnem oblaku obstaja nevarnost podražitev storitev. Prav tako obstaja nevarnost vkleščenja podatkov v posamezen oblak določenega ponudnika. Ko enkrat preselimo večje količine podatkov v javni oblak, je morebitna selitev teh podatkov drugam lahko dolgotrajna in zahtevna. Interoperabilnost med različnimi oblaki je zaenkrat še slaba, saj standardni vmesniki še niso določeni. Prevelika odvisnost od posameznega ponudnika je lahko zelo nevarna, zato moramo ob načrtovanju hibridnega oblaka predvideti možnost povezovanja z alternativnimi ponudniki. V našem primeru lahko relativno enostavno razvijemo nov modul za povezovanje z drugim javnim oblakom. Nikoli se ne smemo popolnoma zanašati le na vire javnega oblaka, saj lahko pride do izpadov delovanja in posledično nedostopnosti virov. V hibridnem oblaku smo za take primere zavarovani z zasebno infrastrukturo. V časih gospodarske negotovosti tudi ni nenavadno, da podjetja propadajo. V primeru, da propade naš ponudnik javnega oblaka, ne moremo več računati na njegove storitve.

Zaključimo lahko, da ima naša rešitev hibridnega oblaka več prednosti, kot slabosti. Rešitev obstoječ zasebni oblak zgolj dopolnjuje z javnim in odločitev o dejanski uporabi virov javnega oblaka prepušča uporabniku.

Poglavje 6

Zaključki in nadaljnje delo

6.1 Zaključki

Namen diplomske naloge je bil rešiti problem preobremenjenosti virtualnega laboratorija ob povečanih potrebah po virih, ki se pojavijo le nekajkrat na leto. Problem smo rešili z integracijo hibridnega oblaka. Najem virov javnega oblaka je stroškovno bolj učinkovit kot nakup nove dodatne infrastrukture, ki bi bila večino časa slabo izkoriščena. V našem primeru, kjer je povečana potreba po virih prisotna ob koncih semestrov in ob rokih za oddajo seminarskih nalog, je hibridna rešitev vsekakor na mestu.

V diplomskem delu smo obravnavali teme računalništva v oblaku in spoznali, da je kljub mladosti področja, le-to že zelo razvito in v mnogih primerih deluje bolje in hitreje kot lastna infrastruktura. Dejstvo ne preseneča, saj so v ozadju velika kapitalska in razvojna vlaganja v to področje, ki je zagotovo prihodnost računalništva. Podjetja, kot je Amazon, so na podlagi lastnih izkušenj zgradila sofisticirano infrastrukturo in ugotovila, da bi jo lahko začela prodajati. Izkušnje, ki si jih je podjetje Amazon pridobilo s spletno trgovino, ki je tipičen predstavnik povečanih potreb po virih v konicah (predpraznična obdobja, razprodaje), je spretno izkoristilo in razvilo nov poslovni model oddajanje računalniške infrastrukture. Sedaj velik del prihodkov predstavlja ravno del podjetja Amazon Web Services (AWS).

Za Laboratorij za računalniške komunikacije smo predlagali arhitekturo hibridnega oblaka, ki zajema obstoječ zasebni oblak z možnostjo povezovanja z javnim oblakom EC2. Za integracijo hibridnega oblaka smo se morali spoznati z okoljem VCL in trenutno namestitvijo zasebnega oblaka v Laboratoriju za računalniške komunikacije. Uspešno smo razvili nov modul za oskrbovanje z viri javnega oblaka EC2, ki služi kot lepilo med zasebnim in javnim oblakom.

Za povezavo z javnim oblakom smo morali preučiti vmesnik API EC2 in izbrati primerno knjižnico, ki nam je olajšala razvoj modula za VCL. Upoštevali smo načela objektno usmerjenega programiranja in dedovanja, s tem smo poizkusili zagotoviti čim boljšo združljivost modulov s prihodnjimi verzijami sistema VCL.

Nov hibridni oblak smo preizkusili v testnem okolju in tako dokazali, da deluje dobro in ima tudi nekaj prednosti pred zasebnim oblakom. Prednosti so predvsem v zaganjalnih časih virtualnih računalnikov, saj se kljub veliki količini istočasnih zahtev po zaganjanju novih virtualnih računalnikov, le-ti hitro vzpostavijo in so pripravljeni na takojšnjo uporabo. V javnem oblaku lahko izvajamo iste funkcionalnosti sistema VCL kot v zasebnem. Te funkcionalnosti zajemajo zaganjanje virtualnih računalnikov, nadziranje njihovega življenjskega cikla z obveščanjem uporabnika o poteku rezervacije in na koncu prisilnim ugašanjem virtualnega računalnika. Prav tako smo implementirali proces zajema nove slike na podlagi obstoječe. Slike iz zasebnega oblaka smo preselili v skladišče slik v javnem oblaku in s tem pospešili delovanje hibridnega oblaka. Z modulom smo dokazali koncept dobrega delovanja hibridnega oblaka v virtualnem laboratoriju.

6.2 Nadaljnje delo in izkušnje

Kot smo že uvodoma omenili, bi lahko hibridno okolje VCL razširili še z najemanjem visoko zmogljivih računalnikov za raziskovalne namene in uporabo javnega oblaka za varnostne kolokacije. Tako bi razširili krog uporabnikov VCL z raziskovalci in ostalim osebjem fakultete. VCL bi služil ko vstopna točka za oskrbovanje z viri, saj omogoča natančno spremljanje uporabe virov (zasebnih in javnih) ter omogoča dodeljevanje različnih nivojev pravic posameznim uporabnikom.

Glede na to, da pri računalništvu v oblaku obstaja možnost 'vkleščenja' v posamezen javni oblak, bi bilo smiselno zgraditi tudi module za povezavo z drugimi javnimi oblaki. Pri tem bi se lahko uporabilo odprtokodne knjižnice (npr. Apache Libcloud ali Apache Deltacloud [17, 18]), ki omogočajo povezovanje z različnimi oblaki na enak način. Treba je poudariti, da so te knjižnice v začetnih fazah razvoja in da se področje stalno spreminja. Prav tako standardi na področju računalništva v oblaku niso natančno definirani in le čas bo pokazal, v katero smer se bo odvijalo.

Na koncu se lahko še vprašamo, ali je smiselno celotno infrastrukturo preseliti v javni oblak? Po našem mnenju zaenkrat ne. Kot prehodno rešitev

priporočamo hibridni oblak, ki ponuja vse prednosti obeh svetov, javnega in zasebnega oblaka.

Možnosti računalništva v oblaku so navidezno neomejene in zanimivo bo videti, kakšno bo stanje na področju čez nekaj let.

Slike

Seznam izvorne kode

Literatura

[1] Michael Ambrust et al., *Above the Coluds: A Berkeley View of Cloud Computing*, UC Berkeley Reliable Adaptive Distributed Systems Laboratory, ZDA, Februar 2009.

[2] Amazon, *Amazon Web Services: Overview of Security Processes*, ZDA, Maj 2011. Dostopno na: http://aws.amazon.com/whitepapers/

[3] Peter Mell, Timothy Grance, *The NIST Definition of Cloud Computing (Draft)*, NIST, ZDA, Januar 2011.

[4] Jithesh Moothoor, Vasvi Bhatt, *A Cloud Computing Solution for Universities: Virtual Computing Lab*, IBM, ZDA, December 2009.

[5] Henry E. Schaffer, Samuel F. Averitt, Marc I. Hoit, Aaron Peeler, Eric D. Sills, Mladen A. Vouk, *"NCSU's Virtual Computing Lab: A Cloud Computing Solution,"* Computer, št. 42, zv. 7, str. 94-97, Julij 2009.

[6] Mladen A. Vouk, Sam Averitt, Michael Bugaev, Andy Kurth, Aaron Peeler, Henry Shaffer, Eric Sills, Sarah Stein, Josh Thompson, *Powered by VCL - Using Virtual Computing Laboratory (VCL)*, v zborniku *Proc. 2nd International Conference on Virtual Computing (ICVCI)*, 15-16 Maj, 2008, str. 1-10.

[7] Mladen A. Vouk, Eric Sills, Patrick Dreher, *Integration of High-Performance Computing into Cloud Computing Services*, v knjigi Furht, Borko; Escalante, Armando (Eds.), *Handbook of Cloud Computing*, ZDA, 2010, pogl. 11.

[8] (2011) Apache VCL - Apache Software Foundation. Dostopno na:
https://cwiki.apache.org/confluence/display/VCL/Apache+VCL

[9] (2011) Amazon Elastic Compute Cloud (EC2) Documentation. Dostopno na:
http://aws.amazon.com/documentation/ec2/

[10] (2011) Amazon EC2 outage: summary and lessons learned. Dostopno na:
http://blog.rightscale.com/2011/04/25/amazon-ec2-outage-summary-and-lessons-learned/

[11] (2011) Amazon EC2 API Reference. Dostopno na:
http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/

[12] (2011) MySQL Community Edition. Dostopno na:
http://www.mysql.com/products/community/

[13] (2011) VM::EC2 knjižnica. Dostopno na:
http://search.cpan.org/~lds/VM-EC2-1.05/

[14] (2011) xCAT Extreme Cloud Administration Toolkit. Dostopno na:
http://xcat.sourceforge.net/

[15] (2011) Summary of the Amazon EC2 and Amazon RDS Service Disruption in the US East Region. Dostopno na:
http://aws.amazon.com/message/65648/

[16] (2011) The Xen hypervisor. Dostopno na:
http://xen.org/

[17] (2011) Apache Libcloud. Dostopno na:
http://libcloud.apache.org/

[18] (2011) Apache Deltacloud. Dostopno na:
http://incubator.apache.org/deltacloud/